Interval Modulation Coding

Saleem Mukhtar and Jehoshua Bruck

Abstract— **In this paper we introduce a new paradigm for storage and communication. We call this paradigm Interval Modulation Coding. Both in the context of communication and storage, one needs to measure the elapsed time between voltage transitions or voltage pulses. Conventionally, this measurement is made by a clock, by counting clock pulses. Analog circuits (or clocks of higher frequency) can also be used to measure elapsed time. And in this case the set of permissible time intervals no longer has to consist of consecutive integer multiples of the clock period but can be chosen in accordance with a probabilistic model of measurement error. We will show that this can potentially provide substantial improvements in terms of bandwitdth and storage density over coding techniques deployed in real storage and communication systems. We provide a mechanism for encoding and decoding data based on variable length to variable length prefix free codes. We show that such codes can be constructed using integer linear programming. From a theoretical standpoint, we study the linear programming relaxation of the integer linear program associated with code construction. We provide an efficient algorithm for determining if the linear programming relaxation is feasible and an efficient algorithm for solving the linear programming relaxation, assuming it is feasible.**

Keywords— **Delay insensitive communication, asynchronous communication, magnetic data storage, frequency modulation, aperiodic frequency modulation, prefix-free codes, prefix codes, variable length to variable length prefix-free codes, variable length to variable length prefix codes, run length limited codes, interval modulation codes, Fibonacci Polyhedra, Generalized Fibonacci Polyhedra, Fibonacci Numbers, Generalized Fibonacci Numbers, linear programming, integer linear programming.**

I. Introduction

The problem of run length limited coding occurs in a variety of contexts which include magnetic storage, holographic storage, wireless and fiber optic communication. In order to motivate the problem via engineering examples, we will start by describing the model for magnetic data storage [30] and asynchronous communication.

A. Model for Magnetic Data Storage

One might think that binary data is stored verbatim on magnetic media like disk drives but practical considerations call for more subtle schemes and these lead directly to the problem described in this paper. We first need to understand how magnetic storage devices such as floppy and hard disk drives actually work. Disk drives contain one or more spinning platters coated with a magnetic medium. Each platter is divided into concentric circular tracks. An electrical read write head floats over the platter on a cushion of air about one hundredth the thickness of a human hair. The head moves radially to access different tracks, but we will confine our attention to a single track.

An electrical current passing through the head will magnetize a piece of the track in effect producing a tiny bar magnet along the track. Reversing the current creates an adjacent bar magnet whose north and south poles are reversed. Data is written by a sequence of current reversals, creating what amounts to a sequence of bar magnets of various lengths and alternating polarity. Data is read by reversing the process. Polarity reversals along the track generate voltage pulses in the head as they pass by. A scheme called peak detection determines when a pulse occurs. By measuring the time elapsed between two adjacent pulses, one can determine the length of the underlying bar magnet. This is illustrated in Figure 1.1.

Due to some practical problems which we describe next, the length of the underlying bar magnets, and consequently the time elapsed between voltage pulses cannot be arbitrary. The first problem is called intersymbol interference. If the polarity changes are too close together, the magnetic lines of force tend to cancel out, and the pulses produced in the head are weaker, and thus harder to detect. Based on the physical characteristics of the magnetic medium and the sensitivity of the electronics, each disk system has a minimum separation distance d between polarity changes for reliable operation.

The other limitations are imposed by the circuitry which drives the read and write process. Conventionally the circuitry used is based on a clock and measurement of elapsed time is made by counting clock pulses. Thus the seperation between voltage pulses must be an integer multiple of the clock period. The third problem is refered to as clock drift. The basic problem being that as the spacing between two pulses increases so does measurement error. This is cor-

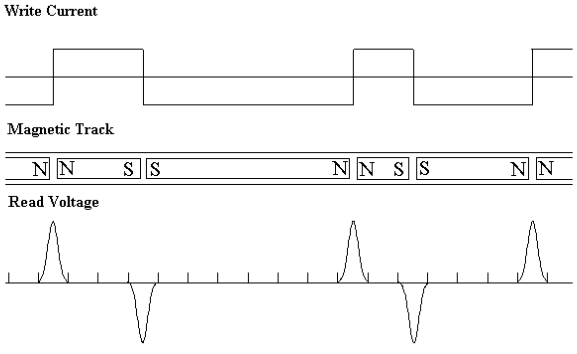


Fig. 1.1. The write current produces a series of bar magnets along the magnetic track. Changes in polarity, produce voltage pulses or spikes.

This work was partially supported by the Lee Center for Advanced Networking at the California Institute of Technology. Saleem Mukhtar is partially supported by a National Defense Science and Engineering Graduate Fellowship.

The authors can be reached via email at saleem@paradise.caltech.edu and bruck@paradise.caltech.edu. Or via normal mail at Mail Code 136-93, California Institute of Technology, Pasadena, CA 91125. USA.

rected by a feedback loop, every time a pulse occurs we can correct for the drift thus preventing it from accumulating to dangerous amounts. Thus we need to ensure that the seperation between two pulses is no more than k units.

These considerations restrict the length of the bar magnets to belong to $\{d, d+1, ..., k-1, k\}$. These bar magnet lengths have to be used to represent a binary sequence. The convention adopted is that a length of l units represents a sequence of $l-1$ 0s followed by a 1. Thus what is stored on the disk is a binary sequence of data which satisfies the following constraints – two adjacent 1s are seperated by at least $d-1$ 0s and there are no more than $k-1$ 0s between two adjacent 1s. These are called run-length limited constraints and the problem of mapping arbitrary binary data to binary sequences that satisfy these constraints has been studied extensively in literature. Adler, et. al. [1] applied techniques from Symbolic Dynamics [30] to produce finite state encoders which could be used for coding and decoding. This work has been extended by Heegard et. al. [20] and Marcus et. al. [33], [31] and [32]. Alternative techniques were introduced by Kautz [25],Franaszek [13] and Immink [22].

B. Model for Asynchronous Communication

A similar problem arises in the context of asynchronous communication. In this context data is encoded in the amplitude of the the signal. Generally $+5V$ is used to represent a logical 1 and $0V$ is used to represent a logical 0. In order to determine the number of 0s or 1s represented by part of the signal between two consecutive voltage transitions, the receiver needs to measure the time between these voltage transitions. Generally this measurement is made by a clock based circuit, by counting clock pulses. If the sequence being transmitted is allowed to contain an arbitrary number of consecutive 0s or an arbitrary number of consecutive 1s then the clock drift can accumulate to dangerous amounts. In order to prevent this from happening, we need to ensure that a voltage tranisition is gauranteed to occur within some fixed amount of time, and the underlying circuitry can use this voltage transition to resynchronize the clock. A commonly used technique is to break up the data to be transmitted into packets where each packet consists of 8 data bits and a start bit. This effectively limits the set of permissible time intervals to be $\{1, 2, ..., 8, 9\}$.

II. New Paradigm

A. Motivation via Rate Analysis

Both in the context of storage and communication one needs to measure elapsed time between voltage pulses or voltage transitions and conventionally this measurement is made by a clock based circuit, by counting clock pulses. Thus the set of permissible time intervals has to consist of consecutive integer multiples of the clock period. Furthermore, since the measurement error increases with elapsed time, one needs to impose a limit on the maximum permissible time between voltage pulses or voltage transitions. Analog circuits (or clocks of higher frequency) can also be used to measure elasped time. In this case the set of permissible time intervals no longer has to consist of consecutive integer multiples of the clock period but can be chosen in accordance with the probabilistic model of measurement error. Before we show that this can potentially yield meaningful improvements in terms of rate and storage density, let us define the maximum achievable rate (or storage density), R^*.

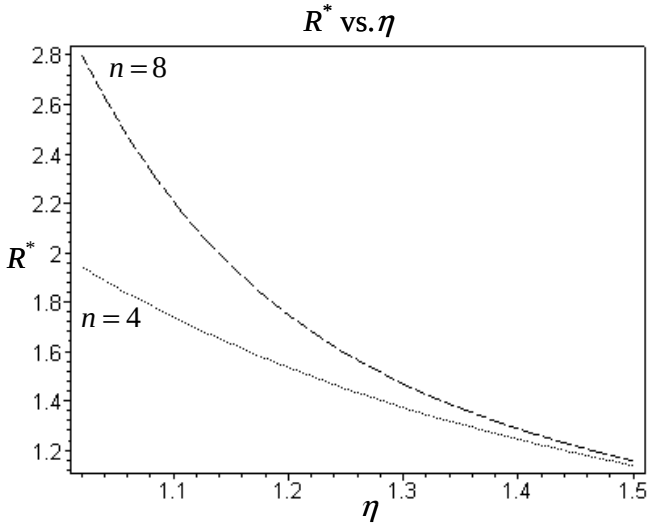


Fig. 2.1. Plot of maximum achievable rate R^* versus η for different number of symbols. (We have assumed measurement error increases linearly with time).

The maximum achievable rate, R^* is defined to be the maximum number of bits that can be transmitted per second. Given a set of permissible time intervals S, let $\bar{N}_S(T)$ be the number of signals or sequences of length less than or equal to T. Then,

$$R^* = \lim_{T \to \infty} \frac{\log_2(\bar{N}_S(T))}{T}$$

Let us assume we use a clock based circuit to measure elapsed time. Furthermore, let us assume the minimum permissible time between voltage pulses or transitions is 1 unit (clock period is 1 unit). Depending on the accuracy of the clock we will have to impose a limit on the maximum permissible time between voltage pulses or voltage transitions. Regardless of the accuracy of the clock, the maximum achievable rate R^* will always be less than or equal to 1.

Now for expository purposes let us assume we have an analog circuit which can be used to measure elapsed time. Furthermore let us assume that the smallest measurable time interval is 1 unit and that measurement error increases linearly with elapsed time. Let δ be the percentage error in measurement and let η be $(1+\delta)/(1-\delta)$. Since we have an analog circuit, the second smallest element in the set of permissible time intervals does not have to be 2 but can be η. Similarly the third smallest element in the set of permissible time intervals no longer has to be 3 but can be η^2 and so on and so forth. Furthermore, we no longer

need to impose a limit on the size of the maximum time interval. The question is how does the maximum achievable rate R^* depend on η and the number of symbols, n. We have plotted the maximum achievable rate R^* as a function of η for some different values of n in Figure 2.1. It is evident that the maximum achievable can be substantially better than 1. In fact as η approaches 1, R^* approaches $\log_2(n)$. This is because as η approaches 1, so does the size of all the symbols. If we have n distinct symbols, each of which has a duration of 1 time unit, then the maximum achievable rate R^* is $\log_2(n)$.

In the context of asynchronous communication the scheme that is commonly used is to break up the data into packets where each packet consists of 8 data bits and a start bit. This effectively limits the size of the symbols to be $\{1, 2, ..., 8, 9\}$. In order to effectively distinguish between an 8 and a 9, the clock drift should be no more than 5.88%. Let us assume we use an analog circuit to measure elapsed time and furthermore, let us assume that the smallest measurable time interval is 1 unit and that the measurement error increases linearly with elapsed time at a rate of 5.88%. Thus $\delta = 1/17 = 5.88\%$ and $\eta = 9/8 = 1.125$. The set of symbols can be $\{1.00, 1.13, 1.27, 1.42, 1.60, 1.80, 2.03, 2.28, ...\}$. In Figure 2.2 we have plotted the maximum achievable rate R^* vs. the number of symbols n for $\eta = 1.125$. From Figure 2.1 and Figure 2.2 it is evident that when $n = 4$ and $\eta = 1.125$ then R^* is 1.68 and when $n = 8$ and $\eta = 1.125$ then R^* is 2.07. Note that the strategy of breaking up the data into packets and inserting start bits cannot achieve a rate of more than 1.

In the context of magnetic data storage the two widely used codes are the $[1, 7]$ code and the $[2, 7]$ code. In the case of the $[1, 7]$ code, the set of permissible time intervals or set of symbols is $\{2, 3, 4, 5, 6, 7, 8\}$. Thus the smallest time interval between two spikes that is measurable by the underlying circuitry is 2 units of time. Furthermore, the circuitry has a measurement error of at most 6.67%. If we assume we have an analog circuit with the same parameters, then a valid set of symbols is $S = \{2.00, 2.29, 2.61, 2.99, 3.41, 3.90, 4.46, 5.09, 5.82, ...\}$. We can show that for $n = 10$, the maximum achievable rate R^* is 1.003. In the $[2, 7]$ code, the set of permissible time intervals or set of symbols is $\{3, 4, 5, 6, 7, 8\}$. Thus the smallest time interval between two spikes that is measurable by the underlying circuitry is 3 units of time. Furthermore, the circuitry has a measurement error of at most 6.67%. If we assume we have an analog circuit with the same parameters, then a valid set of symbols is $S = \{3.00, 3.43, 3.92, 4.48, 5.12, 5.85, 6.68, 7.64, 8.73, ...\}$. We can show that for $n = 10$, the maximum achievable rate R^* is 0.669. The $[2, 7]$ code itself achieves a rate of 0.5.

Hence, both in the context of asynchronous communication and magnetic data storage there are substantial improvements to be realized. What we need is a mechanism for encoding and decoding binary data in a rate efficient manner given an arbitrary set of symbols.

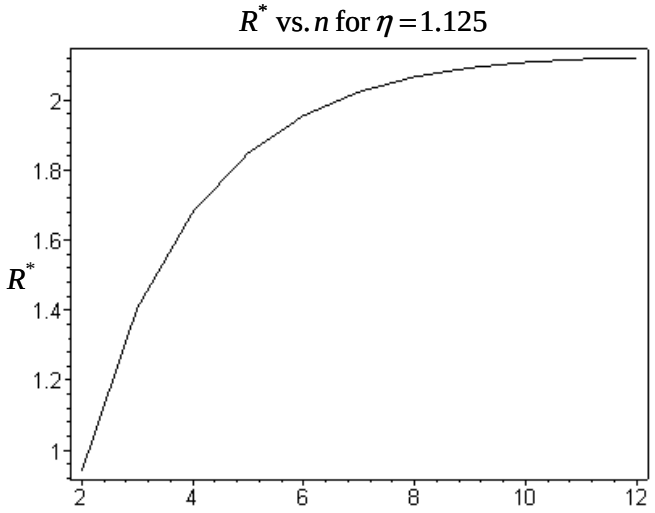


Fig. 2.2. Plot of maximum achievable rate R^* versus n for $\eta = 1.125$. (We have assumed measurement error increases linearly with time).

B. Encoding and Decoding using Variable Length to Variable Length Prefix-Free Codes

For expository purposes consider a simple example. Let the set of permissible time intervals be $\{1, 2\}$. A simple encoding startegy would be to map every binary 0 to a 1 and every binary 1 to a 2. This only achieves a worst case rate of 0.5 since a sequence of T 1s would require $2T$ time units to transmit. An alternate strategy would be to map $00 \rightarrow 111$, $01 \rightarrow 12$, $10 \rightarrow 21$, $110 \rightarrow 112$ and $111 \rightarrow 22$. Note that the sets $\{00, 01, 10, 110, 111\}$ and $\{111, 12, 21, 112, 22\}$ are both prefix-free. In the worst case this scheme would achieve a rate of 0.66 since a sequence of $2T$ 0s would require $3T$ time units to transmit. Since neither the number of bits nor the number of symbols is fixed, this is a variable length to variable length prefix-free code. The code is tabulated in Table 2.1 (D). Notice that the maximum number of bits

A	B	C	D
$R = 0.50$	$R = 0.50$	$R = 0.60$	$R = 0.66$
$0 \leftrightarrow 1$	$00 \leftrightarrow 1111$	$000 \leftrightarrow 11111$	$00 \leftrightarrow 111$
$1 \leftrightarrow 2$	$01 \leftrightarrow 112$	$001 \leftrightarrow 1112$	$01 \leftrightarrow 12$
	$10 \leftrightarrow 121$	$010 \leftrightarrow 1121$	$10 \leftrightarrow 21$
	$11 \leftrightarrow 2$	$011 \leftrightarrow 1211$	$110 \leftrightarrow 112$
		$100 \leftrightarrow 2111$	$111 \leftrightarrow 22$
		$101 \leftrightarrow 122$	
		$110 \leftrightarrow 212$	
		$111 \leftrightarrow 221$	

TABLE 2.1
SOME BLOCK TO VARIABLE LENGTH PREFIX-FREE CODES (A,B,C) AND A VARIABLE LENGTH TO VARIABLE LENGTH PREFIX-FREE CODE (D)

that the encoder might have to examine before it can encode part of the binary sequence is 3 (corresponding to 110 or 111). This is the delay associated with the encoder and is referred to as T_E. Similarly the decoder might have to wait up to 4 units of time before it can map part of the received symbol sequence back to bits (corresponding to 112 or 22). This is the delay associated with the decoder and is referred to as T_D. Furthermore, the size of the codebook is just 5. Given an arbitrary set of permissible time intervals, the delay associated with the encoder T_E, the delay associated with the decoder T_D, and the desired rate R, the problem is to construct the smallest prefix-free code, if one exists. Some codes that can be used for asynchronous communication are shown in Table 2.2. Some codes that can be used in place of the $[1, 7]$ Code are shown in Table 2.3. And some codes that can be used in place of the $[2, 7]$ Code are shown in Table 2.4. These codes provide substantial improvements over coding techniques used in real systems in terms of rate and storage density. Furthermore, these codes are constructed using very few symbols (10) and generally have a small size making them suitable to implement in the form of small lookup tables.

$T_E = 5$ $T_D = 2.73$ $R = 1.83$	$T_E = 6$ $T_D = 3.16$ $R = 1.87$	$T_E = 7$ $T_D = 3.63$ $R = 1.92$
$00 \leftrightarrow S_1$	$00 \leftrightarrow S_1$	$00 \leftrightarrow S_1$
$010 \leftrightarrow S_3$	$010 \leftrightarrow S_3$	$010 \leftrightarrow S_4$
$011 \leftrightarrow S_4$	$011 \leftrightarrow S_4$	$0110 \leftrightarrow S_6$
$100 \leftrightarrow S_5$	$100 \leftrightarrow S_5$	$0111 \leftrightarrow S_7$
$1010 \leftrightarrow S_6$	$1010 \leftrightarrow S_6$	$10000 \leftrightarrow S_2S_1$
$1011 \leftrightarrow S_7$	$1011 \leftrightarrow S_7$	$10001 \leftrightarrow S_2S_2$
$1100 \leftrightarrow S_2S_1$	$1100 \leftrightarrow S_2S_1$	$10010 \leftrightarrow S_3S_1$
$11010 \leftrightarrow S_2S_2$	$11010 \leftrightarrow S_2S_2$	$10011 \leftrightarrow S_8$
$11011 \leftrightarrow S_8$	$11011 \leftrightarrow S_8$	$10100 \leftrightarrow S_2S_3$
$11100 \leftrightarrow S_2S_3$	$11100 \leftrightarrow S_2S_3$	$10101 \leftrightarrow S_3S_2$
$11101 \leftrightarrow S_2S_4$	$11101 \leftrightarrow S_2S_4$	$10110 \leftrightarrow S_3S_3$
$11110 \leftrightarrow S_9$	$11110 \leftrightarrow S_9$	$10111 \leftrightarrow S_2S_4$
$11111 \leftrightarrow S_2S_5$	$111110 \leftrightarrow S_2S_5$	$11000 \leftrightarrow S_9$
	$111111 \leftrightarrow S_2S_7$	$11001 \leftrightarrow S_5S_1$
		$110100 \leftrightarrow S_3S_4$
		$110101 \leftrightarrow S_2S_5$
		$110110 \leftrightarrow S_5S_2$
		$110111 \leftrightarrow S_3S_5$
		$111000 \leftrightarrow S_5S_3$
		$111001 \leftrightarrow S_{10}$
		$111010 \leftrightarrow S_2S_6$
		$111011 \leftrightarrow S_5S_4$
		$111100 \leftrightarrow S_3S_6$
		$1111010 \leftrightarrow S_2S_7$
		$1111011 \leftrightarrow S_5S_5$
		$1111100 \leftrightarrow S_3S_7$
		$1111101 \leftrightarrow S_5S_6$
		$1111110 \leftrightarrow S_3S_8$
		$1111111 \leftrightarrow S_5S_7$

TABLE 2.2
$S = \{1.00, 1.13, 1.27, 1.42, 1.60, 1.80, 2.03, 2.28, 2.57, 2.89\}$
SOME VARIABLE LENGTH TO VARIABLE LENGTH PREFIX-FREE CODES

The prefix-free code construction problems that have been studied in literature are for block to variable length or variable length to block prefix-free codes [29]. In block to variable length prefix-free codes the number of source alphabets is fixed but the number of code alphabets is allowed to vary. In variable length to block prefix-free codes the number of source alphabets is allowed to vary but the number of code alphabets is fixed. Huffman [21] studied the problem of constructing an optimal block to variable length prefix-free code when the probabilities of the source alphabets are different and the transmission costs of the code alphabets are the same. In the codes he constructs the number of code alphabets is allowed to vary, but the number of source alphabets is fixed (to 1). The objective function he minimized was expected transmission time. Varn [46], Perl et. al. [39], Cot [10], Kapoor and Reingold [23] and Golin and Young [17] have studied the problem of constructing an optimal block to variable length prefix-free code when the probabilities of the source alphabets are the same but the transmission costs of the code alphabets are different. Again the objective function they minimized was expected transmission time. Blachman [5], Marcus [34], Karp [24], Krause [28], Cot [11], Mehlhorn [36], Altenkamp and Mehlhorn [2], Gilbert [16] and Golin and Rote [18] have studied the more complex case in which both the source alphabet probabilities as well as the transmission costs of the code alphabets are different. They also minimize expected transmission time. Our problem is most similar to the problem studied in [46], [39], [10], [23] and [17] – since we assume that the probabilities of the source alphabets are the same and the transmission cost of the code alphabets are different. In all these theoretical extensions of Huffman coding, the number of code alphabets is allowed to vary, but the number of source alphabets is fixed to 1. Tunstall [45] pioneered the work on variable length to block prefix-free codes. He allows the number of source alphabets to vary but fixes the number of code alphabets.

The fundamental difference between our work and the work described above is that we use variable length to variable length prefix-free codes. In variable length to variable length prefix-free codes, neither the number of code alphabets nor the number of source alphabets is fixed. Fixing the number of code alphabets or the number of source alphabets adversely affects code size. Reconsider the example illustrated in Table 2.1. Note that the code in Table 2.1 (C) is larger and achieves a lower rate than the code in Table 2.1 (D). In fact, it is easy to show that the smallest block to variable length prefix-free code that achieves a rate of 0.66 has 4096 rules. The variable length to variable length prefix-free code in Table 2.1 (D) has 5 rules and achieves a rate of 0.66.

$T_E = 4$ $T_D = 4.46$ $R = 0.87$	$T_E = 6$ $T_D = 6.75$ $R = 0.89$	$T_E = 7$ $T_D = 7.70$ $R = 0.91$
$00 \leftrightarrow S_1$	$00 \leftrightarrow S_1$	$000 \leftrightarrow S_4$
$01 \leftrightarrow S_2$	$010 \leftrightarrow S_4$	$0010 \leftrightarrow S_6$
$100 \leftrightarrow S_3$	$0110 \leftrightarrow S_6$	$0011 \leftrightarrow S_1S_1$
$101 \leftrightarrow S_4$	$0111 \leftrightarrow S_2S_1$	$0100 \leftrightarrow S_1S_2$
$110 \leftrightarrow S_5$	$1000 \leftrightarrow S_7$	$0101 \leftrightarrow S_2S_1$
$1110 \leftrightarrow S_6$	$10010 \leftrightarrow S_2S_2$	$01100 \leftrightarrow S_1S_3$
$1111 \leftrightarrow S_7$	$10011 \leftrightarrow S_3S_1$	$01101 \leftrightarrow S_3S_1$
	$10100 \leftrightarrow S_2S_3$	$01110 \leftrightarrow S_2S_3$
	$10101 \leftrightarrow S_3S_2$	$01111 \leftrightarrow S_3S_2$
	$10110 \leftrightarrow S_8$	$10000 \leftrightarrow S_1S_4$
	$10111 \leftrightarrow S_3S_3$	$10001 \leftrightarrow S_8$
	$11000 \leftrightarrow S_2S_4$	$10010 \leftrightarrow S_3S_3$
	$11001 \leftrightarrow S_5S_1$	$10011 \leftrightarrow S_2S_4$
	$11010 \leftrightarrow S_3S_4$	$10100 \leftrightarrow S_1S_5$
	$110110 \leftrightarrow S_2S_5$	$10101 \leftrightarrow S_5S_1$
	$110111 \leftrightarrow S_5S_2$	$101100 \leftrightarrow S_3S_4$
	$111000 \leftrightarrow S_9$	$101101 \leftrightarrow S_2S_5$
	$111001 \leftrightarrow S_3S_5$	$101110 \leftrightarrow S_5S_2$
	$111010 \leftrightarrow S_5S_3$	$101111 \leftrightarrow S_9$
	$111011 \leftrightarrow S_2S_6$	$110000 \leftrightarrow S_1S_6$
	$111100 \leftrightarrow S_5S_4$	$110001 \leftrightarrow S_3S_5$
	$111101 \leftrightarrow S_3S_6$	$110010 \leftrightarrow S_5S_3$
	$111110 \leftrightarrow S_{10}$	$110011 \leftrightarrow S_2S_6$
	$111111 \leftrightarrow S_2S_7$	$110100 \leftrightarrow S_5S_4$
		$110101 \leftrightarrow S_1S_7$
		$110110 \leftrightarrow S_7S_1$
		$110111 \leftrightarrow S_3S_6$
		$111000 \leftrightarrow S_2S_2S_1$
		$1110010 \leftrightarrow S_{10}$
		$1110011 \leftrightarrow S_2S_7$
		$1110100 \leftrightarrow S_7S_2$
		$1110101 \leftrightarrow S_5S_5$
		$1110110 \leftrightarrow S_2S_2S_2$
		$1110111 \leftrightarrow S_3S_7$
		$1111000 \leftrightarrow S_7S_3$
		$1111001 \leftrightarrow S_1S_8$
		$1111010 \leftrightarrow S_2S_2S_3$
		$1111011 \leftrightarrow S_5S_6$
		$1111100 \leftrightarrow S_2S_8$
		$1111101 \leftrightarrow S_7S_4$
		$1111110 \leftrightarrow S_2S_2S_4$
		$1111111 \leftrightarrow S_3S_8$

TABLE 2.3
$S = \{2.00, 2.29, 2.61, 2.99, 3.41, 3.90, 4.46, 5.09, 5.82, 6.65\}$
SOME CODES THAT OUTPERFORM THE [1,7] CODE

$T_E = 4$ $T_D = 6.68$ $R = 0.58$	$T_E = 6$ $T_D = 10.11$ $R = 0.59$	$T_E = 8$ $T_D = 12.76$ $R = 0.61$
$00 \leftrightarrow S_1$	$00 \leftrightarrow S_1$	$00 \leftrightarrow S_1$
$01 \leftrightarrow S_2$	$010 \leftrightarrow S_4$	$010 \leftrightarrow S_4$
$100 \leftrightarrow S_3$	$0110 \leftrightarrow S_6$	$0110 \leftrightarrow S_6$
$101 \leftrightarrow S_4$	$0111 \leftrightarrow S_2S_1$	$0111 \leftrightarrow S_2S_1$
$110 \leftrightarrow S_5$	$1000 \leftrightarrow S_7$	$10000 \leftrightarrow S_2S_2$
$1110 \leftrightarrow S_6$	$10010 \leftrightarrow S_2S_2$	$10001 \leftrightarrow S_3S_1$
$1111 \leftrightarrow S_7$	$10011 \leftrightarrow S_3S_1$	$10010 \leftrightarrow S_2S_3$
	$10100 \leftrightarrow S_2S_3$	$10011 \leftrightarrow S_3S_2$
	$10101 \leftrightarrow S_3S_2$	$10100 \leftrightarrow S_8$
	$10110 \leftrightarrow S_8$	$10101 \leftrightarrow S_3S_3$
	$10111 \leftrightarrow S_3S_3$	$10110 \leftrightarrow S_2S_4$
	$11000 \leftrightarrow S_2S_4$	$10111 \leftrightarrow S_5S_1$
	$11001 \leftrightarrow S_5S_1$	$110000 \leftrightarrow S_3S_4$
	$11010 \leftrightarrow S_3S_4$	$110001 \leftrightarrow S_2S_5$
	$110110 \leftrightarrow S_2S_5$	$110010 \leftrightarrow S_5S_2$
	$110111 \leftrightarrow S_5S_2$	$110011 \leftrightarrow S_9$
	$111000 \leftrightarrow S_9$	$110100 \leftrightarrow S_3S_5$
	$111001 \leftrightarrow S_3S_5$	$110101 \leftrightarrow S_5S_3$
	$111010 \leftrightarrow S_5S_3$	$110110 \leftrightarrow S_2S_6$
	$111011 \leftrightarrow S_2S_6$	$110111 \leftrightarrow S_5S_4$
	$111100 \leftrightarrow S_5S_4$	$111000 \leftrightarrow S_7S_1$
	$111101 \leftrightarrow S_3S_6$	$111001 \leftrightarrow S_3S_6$
	$111110 \leftrightarrow S_{10}$	$1110100 \leftrightarrow S_{10}$
	$111111 \leftrightarrow S_2S_7$	$1110101 \leftrightarrow S_2S_7$
		$1110110 \leftrightarrow S_7S_2$
		$1110111 \leftrightarrow S_5S_5$
		$1111000 \leftrightarrow S_3S_7$
		$1111001 \leftrightarrow S_7S_3$
		$1111010 \leftrightarrow S_5S_6$
		$1111011 \leftrightarrow S_2S_8$
		$1111100 \leftrightarrow S_7S_4$
		$11111010 \leftrightarrow S_3S_8$
		$11111011 \leftrightarrow S_5S_7$
		$11111100 \leftrightarrow S_7S_5$
		$11111101 \leftrightarrow S_2S_9$
		$11111110 \leftrightarrow S_3S_9$
		$11111111 \leftrightarrow S_5S_8$

TABLE 2.4
$S = \{3.00, 3.43, 3.92, 4.48, 5.12, 5.85, 6.68, 7.64, 8.73, 9.98\}$
SOME CODES THAT OUTPERFORM THE [2,7] CODE

III. CONTRIBUTION AND ORGANIZATION

In this paper we have introduced a new paradigm for communication and storage. We call this paradigm Interval Modulation Coding. Both in the context of storage and communication, one needs to measure elapsed time between voltage pulses or voltage transitions. Conventionally this measurement is made by a clock based circuit, by counting clock pulses. The key idea is that analog circuits (or clocks of higher frequency) can be used to measure elapsed time and in this case the set of permissible time intervals can be chosen in accordance with a probabilistic model of measurement error. We have shown that using this idea one can potentially realize substantial improvements in terms of rate and storage density. We have also provided a mechanism for encoding and decod-

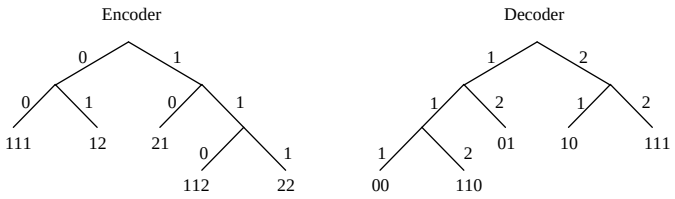


Fig. 4.1. Encoder and Decoder for Code in Table 2.1 (D) Implemented using Prefix Trees

ing data based on variable length to variable length prefix-free codes. In Section IV we formally define the code construction problem and show that optimal variable length to variable length prefix free codes can be constructed using integer linear programming. We also define the linear programming relaxation of the integer linear program and formulate a sufficient and necessary condition for the linear programming relaxation to have a solution. In Section V we formally define a Generalized Fibonacci Polyhedra. This definition is motivated by the structure of the linear programming problem associated with code construction. We show that such polyhedra satisfy some special properties. In Section VI we further build on these properties and present an efficient algorithm for maximizing linear functions over a Generalized Fibonacci Polyhedra. We also present an efficient algorithm for minimizing a linear function over the intersection of a Generalized Fibonacci Polyhedra and a hyperplane. In Section VII we illustrate how the algorithms developed in this paper can be used to determine if the linear programming relaxation of the integer program associated with code construction has a solution, and find the optimal solution, if one exists. We will also illustrate how one can construct codes, given a solution to the integer program associated with code construction. In Section VIII we derive closed form expressions for Generalized Fibonacci Numbers and their sums. The work on Generalized Fibonacci Numbers is an extension of the work of Capocelli and Cull [9].

IV. Problem Formulation and Code Construction Algorithm

Let us call the set of permissible time intervals or symbols, S. We assume that the set S only contains integers[1]. Furthermore, we assume that S has at least 2 elements. Note that variable length to variable length prefix-free codes can be implemented using prefix trees. We have implemented the code in Table 2.1 (D) in Figure 4.1.

Definition 4.1: For each leaf node x of the encoder tree T, let $d_e(x)$ denote the length of the path from the root to x, called the "encoder delay" of x. Also, let $d_d(x) = x_1 + ... + x_n$, where $x_1, ..., x_n$ is in S^n is the label of x, called the "decoder delay" of x.

Let $\max_x d_e(x)$ and let $\max_x d_d(x)$ be called the "maximum encoder"and "maximum decoder" delays, respectively, taken over all leaf nodes x of the tree T. We will refer to these as T_E and T_D respectively. Also define the "rate" to be $R = \min_x d_e(x)/d_d(x)$.

Problem 4.1: Given numbers T_E, T_D, and R, and a set of positive integers S, find a tree T with the fewest possible leaves such that the maximum encoder delay is T_E, the maximum decoder delay is T_D, the rate is at least R, and the labels of the leaves form a prefix-free set over S.

Let m be the largest element in S. For $i \in \{1, 2, ..., m-1, m\}$, let

$$K_i = \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases} \tag{1}$$

We define $N_K(T)$ to be the number of sequences of length T whose elements are in S.

$$N_K(T) = \begin{cases} \sum_{i=1}^{m} K_i N_K(T-i) & T > m \\ K_T + \sum_{i=1}^{T-1} K_i N_K(T-i) & 2 \le T \le m \\ K_1 & T = 1 \end{cases} \tag{2}$$

It is easy to show that $N_K(T)$ can be computed using the recurrence above. When $T = 1$, $N_K(T)$ is 1 if $K_1 = 1$ ($1 \in S$) and 0 if $K_1 = 0$ ($0 \notin S$). Hence, when $T = 1$, $N_K(T) = K_1$. Now consider the case when $2 \le T \le m$. The number of sequences of length T that end in T is K_T (1 if $T \in S$, 0 if $T \notin S$). The number of sequences of length T that end in i is $K_i N_K(T-i)$ ($N_K(T-i)$ if $T \in S$, 0 if $T \notin S$). Hence, $N_K(T) = K_T + \sum_{i=1}^{T-1} K_i N_K(T-i)$. The proof for when $T > m$ is similar and left as an exercise for the reader. It is easy to show that N_K is a generalization of the Fibonacci Numbers. If $S = \{1, 2, ..., m-1, m\}$, $N_K(T)$ is the $(T+1)$th Fibonacci Number of order m [9].

Example 4.1: First consider the case when $S = \{1, 2, 3\}$. In this case $m = 3$ and $K_1 = 1$, $K_2 = 1$ and $K_3 = 1$. Using the recurrence above, we find $N_K(5) = 13$. The 13 sequences of length 5 are $\{11111, 1112, 1121, 1211, 2111, 122, 212, 221, 113, 131, 311, 23, 32\}$. Now consider the case when $S = \{2, 3, 5\}$. In this case $m = 5$ and $K_1 = 0$, $K_2 = 1$, $K_3 = 1$, $K_4 = 0$ and $K_5 = 1$. Using the recurrence above we find $N_K(9) = 8$. The 8 sequences of length 9 are $\{2223, 2232, 2322, 3222, 333, 225, 252, 522\}$.

For $l \in \{1, 2, ..., T_D\}$ and $d \in \{1, 2, ..., T_E\}$, let x_d^l be the number of leafs (non negative integer) in the encoder tree at depth d which have a label of length l. Also, we define x^l to be the number of labels of length l and we define x_d to be the number of leafs in the encoder at depth d. Formally,

$$x^l = \sum_{i=1}^{T_E} x_i^l \text{ and } x_d = \sum_{i=1}^{T_D} x_d^i \tag{3}$$

The problem is to design the smallest encoder decoder pair. So we would like to,

$$\min \sum_{i=1}^{T_D} \sum_{j=1}^{T_E} x_j^i \tag{4}$$

[1]If the symbols are not integers, they must be suitably scaled and truncated or rounded.

Firstly, since the encoder tree is a full prefix tree, the Kraft Inequality for full trees, tells us that

$$\sum_{i=1}^{T_E} 2^{(T_E-i)} x_i = 2^{T_E} \tag{5}$$

Furthermore the labels attached to the leaf nodes of the encoder tree must form a prefix-free set over S. Equivalently, the decoder must be a prefix tree. From Theorem 10.1 (see appendix) we know that,

$$\text{For } 1 \leq L \leq T_D,\ x^L + \sum_{l=1}^{L-1} N_K(L-l)x^l \leq N_K(L) \tag{6}$$

Also the encoder and decoder must achieve a rate of R. Hence,

$$j/i < R \Longrightarrow x_j^i = 0 \tag{7}$$

It is easy to verify that equations (5,6,7) are also sufficient. If we are given x_j^i which satisfy these constraints we can construct a code which achieves the desired rate and has the specified delays. Since equation (5) is satisfied, the Kraft inequality for full binary trees tells us that we can construct a full binary tree such that the number of leafs at depth i is x_i. Since equation (6) is satisfied, Theorem 10.1 tells us that we can find a prefix-free set M_S over S such that the number of labels in M_S of length l is x^l. We can now arbitrarily assign x_j^i labels of length i from the set M_S to leaf nodes of the encoder tree at depth j in a one on one manner.

Equations (4,5,6,7) represent an integer linear program. Equations (4,5,6,7) without the integrality constraint are the linear programming relaxation of the integer linear program. Although the solution to the linear programming relaxation does not provide a code, it does provide a lower bound on the size of the optimal code. In this paper we will present a polynomial time[2] algorithm for finding an optimal solution to the linear programming relaxation, if one exists. We next formulate a necessary and sufficient condition for the linear programming relaxation to have a solution.

Note that equation (5) represents a hyperplane and equations (6,7) represent a convex polyhedra. For a solution to exist, this hyperplane must intersect the convex polyhedra. It must be noted that this criteria is both necessary and sufficient for the linear programming relaxation of the integer linear program to have a solution. However, it is not sufficient, only necessary to ensure the integer linear program has a solution. The hyperplane intersects that polyhedra if and only if $\max \sum_{i=1}^{T_E} 2^{(T_E-i)} x_i$ subject to equations (6,7) is greater than or equal to 2^{T_E}. In this paper we also present a polynomial time algorithm for performing this optimization.

[2]We define a polynomial time algorithm to be one that requires a polynomial number of arithmetic operations, integer or floating point.

V. Generalized Fibonacci Polyhedra and their Properties

First we will define a Generalized Fibonacci Polyhedra. This definition is motivated by the matrix representation of the system of inequalities in equation (6).

Notation 5.1: We let I^n be the $n \times n$ identity matrix and we let A^n be the $n \times n$ matrix below,

$$A^n = \begin{bmatrix} 1 & N_K(1) & . & N_K(n-2) & N_K(n-1) \\ 0 & 1 & . & N_K(n-3) & N_K(n-2) \\ . & . & . & . & . \\ 0 & 0 & . & 1 & N_K(1) \\ 0 & 0 & . & 0 & 1 \end{bmatrix}$$

We refer to the ith column of A^n as A_i^n and the ith column of I^n as I_i^n. Note that A_i^n can be computed recursively as follows,

$$A_i^n = \begin{cases} I_i^n + \sum_{j=1}^{m} K_j A_{i-j}^n & i > m \\ I_i^n + \sum_{j=1}^{i-1} K_j A_{i-j}^n & 2 \leq i \leq m \\ I_1^n & i = 1 \end{cases}$$

Definition 5.1: We are given $K \in \{0,1\}^m$ and $\alpha \in \{0, R^+\}^n$. Let $\Gamma(K,\alpha) = \{y | y \in \{0,R^+\}^n$ and $A^n y \leq A^n \alpha\}$. Equivalently if $s \in \{0,R^+\}^n$ is the vector of slack variables we have $\Gamma(K,\alpha) = \{y | y, s \in \{0,R^+\}^n$ and $A^n y + I^n s = A^n \alpha\}$. We will refer to $\Gamma(K,\alpha)$ as a Generalized Fibonacci Polyhedra of dimension n and order m.

A. Preliminaries

Lemma 5.1: We are given $K \in \{0,1\}^m$, $\alpha \in \{0,R^+\}^n$ and $y \in \Gamma(K,\alpha)$. Let s be the vector of slack variables corresponding to y.

If $2 \leq n \leq m$,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} = \sum_{i=1}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}$$

If $n > m$,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} = \sum_{i=1}^{n-m-1} \alpha_i A_i^{n-1} + \sum_{i=n-m}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}$$

Proof: Since $y \in \Gamma(K,\alpha)$, $Ay + Is = A\alpha$. Hence,

$$\sum_{i=1}^{n} y_i A_i^n + \sum_{i=1}^{n} s_i I_i^n = \sum_{i=1}^{n} \alpha_i A_i^n$$

Note that $y_n + s_n = \alpha_n$. Hence,

$$\sum_{i=1}^{n-1} y_i A_i^n + \sum_{i=1}^{n-1} s_i I_i^n = \sum_{i=1}^{n-1} \alpha_i A_i^n + \alpha_n A_n^n - y_n A_n^n - s_n I_n^n$$
$$= \sum_{i=1}^{n-1} \alpha_i A_i^n + (\alpha_n - y_n)(A_n^n - I_n^n)$$

Now consider the case when $2 \leq n \leq m$. In this case $A_n^n = I_n^n + \sum_{i=1}^{n-1} K_i A_{n-i}^n$. Hence,

$$\sum_{i=1}^{n-1} y_i A_i^n + \sum_{i=1}^{n-1} s_i I_i^n$$
$$= \sum_{i=1}^{n-1} \alpha_i A_i^n + (\alpha_n - y_n)(I_n^n + \sum_{i=1}^{n-1} K_i A_{n-i}^n - I_n^n)$$
$$= \sum_{i=1}^{n-1} \alpha_i A_i^n + (\alpha_n - y_n) \sum_{i=1}^{n-1} K_{n-i} A_i^n$$
$$= \sum_{i=1}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^n$$

Hence,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} = \sum_{i=1}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}$$

If $n > m$, $A_n^n = I_n^n + \sum_{i=1}^{m} K_i A_{n-i}^n$. We can show,

$$\sum_{i=1}^{n-1} y_i A_i^n + \sum_{i=1}^{n-1} s_i I_i^n$$
$$= \sum_{i=1}^{n-1} \alpha_i A_i^n + (\alpha_n - y_n) \sum_{i=n-m}^{n-1} K_{n-i} A_i^n$$
$$= \sum_{i=1}^{n-m-1} \alpha_i A_i^n + \sum_{i=n-m}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^n$$

Hence,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1}$$
$$= \sum_{i=1}^{n-m-1} \alpha_i A_i^{n-1} + \sum_{i=n-m}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1} \quad \blacksquare$$

B. The 0-1 Principle

Before we prove the 0-1 Principle, we will introduce some notation and prove a lemma.

Definition 5.2: Let $a, b \in \{0, R^+\}^n$. We say that $a \subseteq b$ iff for all $i \in \{1, ..., n\}$, $a_i > 0$ implies $b_i > 0$.

Lemma 5.2: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$ and $y \in \Gamma(K, \alpha)$. If we are given $\bar{\alpha} \in \{0, R^+\}^n$ such that $\bar{\alpha} \subseteq \alpha$, then we can find $\bar{y} \in \Gamma(K, \bar{\alpha})$ such that $\bar{y} \subseteq y$ and $\bar{s} \subseteq s$, where s is the vector of slack variables corresponding to y and $\bar{s}$ is the vector of slack variables corresponding to $\bar{y}$.

Proof: The proof is by induction.

Base Case: $n = 1$. We are given $y_1 + s_1 = \alpha_1$. If $\alpha_1 = 0$ then $y_1 = 0$, $s_1 = 0$ and $\bar{\alpha}_1 = 0$. We let $\bar{y}_1 = 0$ and $\bar{s}_1 = 0$. Note that $\bar{y}_1 + \bar{s}_1 = \bar{\alpha}_1$ and $\bar{y} \subseteq y$ and $\bar{s} \subseteq s$. If $\alpha_1 > 0$ then let $\bar{y}_1 = \bar{\alpha}_1 y_1 / \alpha_1$ and $\bar{s}_1 = \bar{\alpha}_1 s_1 / \alpha_1$. Note that $\bar{y}_1 + \bar{s}_1 = \bar{\alpha}_1 y_1 / \alpha_1 + \bar{\alpha}_1 s_1 / \alpha_1 = \bar{\alpha}_1 (y_1 + s_1) / \alpha_1 = \bar{\alpha}_1$. Furthermore $\bar{y}_1 > 0$ implies $y_1 > 0$ and $\bar{s}_1 > 0$ implies $s_1 > 0$. Hence, $\bar{y} \subseteq y$ and $\bar{s} \subseteq s$.

Inductive Hypothesis: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^{n-1}$ and $y \in \Gamma(K, \alpha)$. If we are given $\bar{\alpha} \in \{0, R^+\}^{n-1}$ such that $\bar{\alpha} \subseteq \alpha$, then we can find $\bar{y} \in \Gamma(K, \bar{\alpha})$ such that $\bar{y} \subseteq y$ and $\bar{s} \subseteq s$, where s is the vector of slack variables corresponding to y and $\bar{s}$ is the vector of slack variables corresponding to $\bar{y}$.

We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$ and $y \in \Gamma(K, \alpha)$. Let s be the vector of slack variables corresponding to y. We will assume that $\alpha_n > 0$. If $\alpha_n = 0$ then $y_n = 0$ and $s_n = 0$ and the result follows trivially from the induction hypothesis. There are two cases to consider.

Case 1 : $2 \leq n \leq m$. Since $y \in \Gamma(K, \alpha)$, by Lemma 5.1,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} = \sum_{i=1}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}$$

We let $\bar{y}_n = \bar{\alpha}_n y_n / \alpha_n$ and $\bar{s}_n = \bar{\alpha}_n s_n / \alpha_n$ (Note that $\bar{y}_n > 0$ implies $y_n > 0$ and $\bar{s}_n > 0$ implies $s_n > 0$). Next we show that $\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i} > 0$ implies $\alpha_i + (\alpha_n - y_n) K_{n-i} > 0$. If $\bar{\alpha}_n - \bar{y}_n > 0$ then $\alpha_n - y_n > 0$. Hence, if $(\bar{\alpha}_n - \bar{y}_n) K_{n-i} > 0$ then $(\alpha_n - y_n) K_{n-i} > 0$. Since $\bar{\alpha}_i > 0$ implies $\alpha_i > 0$, we have $\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i} > 0$ implies $\alpha_i + (\alpha_n - y_n) K_{n-i} > 0$. By the inductive hypothesis we can find $\bar{y}_1, ..., \bar{y}_{n-1}$ and $\bar{s}_1, ..., \bar{s}_{n-1}$ such that $\bar{y}_i > 0$ implies $y_i > 0$ and $\bar{s}_i > 0$ implies $s_i > 0$ and,

$$\sum_{i=1}^{n-1} \bar{y}_i A_i^{n-1} + \sum_{i=1}^{n-1} \bar{s}_i I_i^{n-1} = \sum_{i=1}^{n-1} [\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i}] A_i^{n-1}$$

$$\sum_{i=1}^{n-1} \bar{y}_i A_i^n + \sum_{i=1}^{n-1} \bar{s}_i I_i^n = \sum_{i=1}^{n-1} [\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i}] A_i^n$$
$$= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \sum_{i=1}^{n-1} [(\bar{\alpha}_n - \bar{y}_n) K_i] A_{n-i}^n$$
$$= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + (\bar{\alpha}_n - \bar{y}_n)(A_n^n - I_n^n)$$

Hence,

$$\sum_{i=1}^{n} \bar{y}_i A_i^n + \sum_{i=1}^{n} \bar{s}_i I_i^n$$
$$= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + (\bar{\alpha}_n - \bar{y}_n)(A_n^n - I_n^n) + \bar{y}_n A_n^n + \bar{s}_n I_n^n$$
$$= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \bar{s}_n (A_n^n - I_n^n) + \bar{y}_n A_n^n + \bar{s}_n I_n^n$$
$$= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \bar{s}_n A_n^n + \bar{y}_n A_n^n$$
$$= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \bar{\alpha}_n A_n^n = \sum_{i=1}^{n} \bar{\alpha}_i A_i^n$$

Case 2 : $n > m$. Since $y \in \Gamma(K, \alpha)$, by Lemma 5.1,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} = \sum_{i=1}^{n-m-1} \alpha_i A_i^{n-1} + \sum_{i=n-m}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}$$

We let $\bar{y}_n = \bar{\alpha}_n y_n / \alpha_n$ and $\bar{s}_n = \bar{\alpha}_n s_n / \alpha_n$ (Note that $\bar{y}_n > 0$ implies $y_n > 0$ and $\bar{s}_n > 0$ implies $s_n > 0$). For $i \in \{1, ..., n-m-1\}$ $\bar{\alpha}_i > 0$ implies $\alpha_i > 0$. Next we show that for $i \in \{n-m, ..., n\}$ $\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i} > 0$ implies $\alpha_i + (\alpha_n - y_n) K_{n-i} > 0$. If $\bar{\alpha}_n - \bar{y}_n > 0$ then $\alpha_n - y_n > 0$. Hence, if $(\bar{\alpha}_n - \bar{y}_n) K_{n-i} > 0$ then $(\alpha_n - y_n) K_{n-i} > 0$. Since $\bar{\alpha}_i > 0$ implies $\alpha_i > 0$, we have $\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i} > 0$ implies $\alpha_i + (\alpha_n - y_n) K_{n-i} > 0$. By the inductive hypothesis we can find $\bar{y}_1, ..., \bar{y}_{n-1}$and $\bar{s}_1, ..., \bar{s}_{n-1}$ such that for $i \in \{1, ..., n-1\}$ $\bar{y}_i > 0$ implies $y_i > 0$ and $\bar{s}_i > 0$ implies $s_i > 0$ and,

$$\sum_{i=1}^{n-1} \bar{y}_i A_i^{n-1} + \sum_{i=1}^{n-1} \bar{s}_i I_i^{n-1} = \sum_{i=1}^{n-m-1} \bar{\alpha}_i A_i^{n-1} + \sum_{i=n-m}^{n-1} [\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i}] A_i^{n-1}$$

$$\begin{aligned}
&\sum_{i=1}^{n-1} \bar{y}_i A_i^n + \sum_{i=1}^{n-1} \bar{s}_i I_i^n \\
&= \sum_{i=1}^{n-m-1} \bar{\alpha}_i A_i^n + \sum_{i=n-m}^{n-1} [\bar{\alpha}_i + (\bar{\alpha}_n - \bar{y}_n) K_{n-i}] A_i^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \sum_{i=n-m}^{n-1} (\bar{\alpha}_n - \bar{y}_n) K_{n-i} A_i^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \sum_{i=1}^{m} [(\bar{\alpha}_n - \bar{y}_n) K_i] A_{n-i}^n
\end{aligned}$$

Hence,

$$\begin{aligned}
&\sum_{i=1}^{n} \bar{y}_i A_i^n + \sum_{i=1}^{n} \bar{s}_i I_i^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \sum_{i=1}^{m} [(\bar{\alpha}_n - \bar{y}_n) K_i] A_{n-i}^n + \bar{y}_n A_n^n + \bar{s}_n I_n^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + (\bar{\alpha}_n - \bar{y}_n)(A_n^n - I_n^n) + \bar{y}_n A_n^n + \bar{s}_n I_n^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \bar{s}_n (A_n^n - I_n^n) + \bar{y}_n A_n^n + \bar{s}_n I_n^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \bar{s}_n A_n^n + \bar{y}_n A_n^n \\
&= \sum_{i=1}^{n-1} \bar{\alpha}_i A_i^n + \bar{\alpha}_n A_n^n = \sum_{i=1}^{n} \bar{\alpha}_i A_i^n
\end{aligned}$$

■

Now we are ready to state and prove the 0-1 Principle.

Theorem 5.1: We are given $K \in \{0,1\}^m$ and $\alpha \in \{0, R^+\}^n$. If $y \in \{0, R^+\}^n$ is an extreme point of $\Gamma(K, \alpha)$ and $s \in \{0, R^+\}^n$ is the vector of slack variables then $(y_n, s_n) = (0, \alpha_n)$ or $(y_n, s_n) = (\alpha_n, 0)$.

Proof: The proof is by contradiction. Assume y is an extreme point of $\Gamma(K, \alpha)$ and $s \in \{0, R^+\}^n$ is the corresponding vector of slack variables. Since y is an extreme point of $\Gamma(K, \alpha)$, $y \in \Gamma(K, \alpha)$. Hence $A^n y \leq A^n \alpha$ or equivalently $A^n y + I^n s = A^n \alpha$. Let V be the set of vectors that are used by y, s. Specifically $A_i^n \in V$ iff $y_i > 0$ and $I_i^n \in V$ iff $s_i > 0$. Since $A^n y + I^n s = A^n \alpha$, $y_n + s_n = \alpha_n$. Furthermore, $y_n \geq 0$ and $s_n \geq 0$. Hence $0 \leq y_n \leq \alpha_n$. If $\alpha_n = 0$ then $(y_n, s_n) = (0, 0)$. So we will assume $\alpha_n > 0$. Now assume $(y_n, s_n) \neq (0, \alpha_n)$ and $(y_n, s_n) \neq (\alpha_n, 0)$. Hence $0 < y_n < \alpha_n$ and $s_n = \alpha_n - y_n > 0$. We will show that if this is the case then the vectors in V are not linearly independent and consequently y is not an extreme point of $\Gamma(K, \alpha)$.

Case 1 : $n = 1$. We have assumed $\alpha_1 > 0$, $0 < y_1 < \alpha_1$ and $s_1 = \alpha_1 - y_1 > 0$. Since $y_1 > 0$, $A_1^1 \in V$. Since $s_1 > 0$, $I_1^1 \in V$. Since $A_1^1 \in V$, $I_1^1 \in V$ and $A_1^1 = I_1^1$, V cannot be a linearly independent set of vectors.

Case 2 : $2 \leq n \leq m$. By Lemma 5.1 we know that,

$$\sum_{i=1}^{n-1} y_i A_i^n + \sum_{i=1}^{n-1} s_i I_i^n = \sum_{i=1}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^n$$

We have assumed $\alpha_n > 0$, $0 < y_n < \alpha_n$ and $s_n = \alpha_n - y_n > 0$. Since $y_n > 0$, $A_n^n \in V$. Since $s_n > 0$, $I_n^n \in V$. Now note that if $K_{n-i} > 0$ then $\alpha_i + (\alpha_n - y_n) K_{n-i} > 0$. This is because $(\alpha_n - y_n) > 0$ and $\alpha_i \geq 0$. By Lemma 5.2 we can find $\bar{y} \in \{0, R^+\}^{n-1}$ and the vector of slack variables $\bar{s} \in \{0, R^+\}^{n-1}$ such that for $i \in \{1, ..., n-1\}$ $\bar{y}_i > 0$ implies $y_i > 0$ and $\bar{s}_i > 0$ implies $s_i > 0$. Furthermore,

$$\sum_{i=1}^{n-1} \bar{y}_i A_i^n + \sum_{i=1}^{n-1} \bar{s}_i I_i^n = \sum_{i=1}^{n-1} K_{n-i} A_i^n = \sum_{i=1}^{n-1} K_i A_{n-i}^n$$

Note that $I_n^n \in V$ and

$$I_n^n + \sum_{i=1}^{n-1} \bar{y}_i A_i^n + \sum_{i=1}^{n-1} \bar{s}_i I_i^n = I_n^n + \sum_{i=1}^{n-1} K_i A_{n-i}^n = A_n^n$$

Hence, by using a subset of the vectors in V excluding A_n^n we have been able to generate A_n^n which is also in V. Hence the vectors in V are not linearly independent.

Case 3 : $n > m$. By Lemma 5.1 we know that,

$$\sum_{i=1}^{n-1} y_i A_i^n + \sum_{i=1}^{n-1} s_i I_i^n = \sum_{i=1}^{n-m-1} \alpha_i A_i^n + \sum_{i=n-m}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^n$$

We have assumed $\alpha_n > 0$, $0 < y_n < \alpha_n$ and $s_n = \alpha_n - y_n > 0$. Since $y_n > 0$, $A_n^n \in V$. Since $s_n > 0$, $I_n^n \in V$. Now note that if $K_{n-i} > 0$ then $\alpha_i + (\alpha_n - y_n) K_{n-i} > 0$. This is because $(\alpha_n - y_n) > 0$ and $\alpha_i \geq 0$. By Lemma 5.2 we

can find $\bar{y} \in \{0, R^+\}^{n-1}$ and the vector of slack variables $\bar{s} \in \{0, R^+\}^{n-1}$ such that for $i \in \{1, ..., n-1\}$ $\bar{y}_i > 0$ implies $y_i > 0$ and $\bar{s}_i > 0$ implies $s_i > 0$. Furthermore,

$$\sum_{i=1}^{n-1} \bar{y}_i A_i^n + \sum_{i=1}^{n-1} \bar{s}_i I_i^n = \sum_{i=n-m}^{n-1} K_{n-i} A_i^n = \sum_{i=1}^{m} K_i A_{n-i}^n$$

Note that $I_n^n \in V$ and

$$I_n^n + \sum_{i=1}^{n-1} \bar{y}_i A_i^n + \sum_{i=1}^{n-1} \bar{s}_i I_i^n = I_n^n + \sum_{i=1}^{m} K_i A_{n-i}^n = A_n^n$$

Hence, by using a subset of the vectors in V excluding A_n^n we have been able to generate A_n^n which is also in V. Hence the vectors in V are not linearly independent. ■

C. The Decomposition Principle

Lemma 5.3: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$ and $y \in \Gamma(K, \alpha)$. For $i \in \{1, ..., k\}$, we are given $\alpha^i \in \{0, R^+\}^n$ such that $\sum_{i=1}^k \alpha^i = \alpha$ then we can find $y^i \in \Gamma(K, \alpha^i)$ such that $\sum_{i=1}^k y^i = y$ and $\sum_{i=1}^k s^i = s$ where s is the vector of slack variables corresponding to y and s^i is the vector of slack variables corresponding to y^i.

Proof: The proof is by induction.

Base Case: $n = 1$. We are given $y_1 + s_1 = \alpha_1$. If $\alpha_1 = 0$ then $y_1 = 0$ and $s_1 = 0$. Futhermore for all $i \in \{1, ..., k\}$ $\alpha^i = 0$ since $\sum_{i=1}^k \alpha^i = \alpha = 0$ and $\alpha^i \in \{0, R^+\}^n$. For all $i \in \{1, ..., k\}$ we let $y_1^i = 0$ and $s_1^i = 0$. Note that $y_1^i + s_1^i = \alpha_1^i$ and $\sum_{i=1}^k y_1^i = y_1$ and $\sum_{i=1}^k s_1^i = s_1$. If $\alpha_1 > 0$ then we let $y_1^i = \alpha_1^i y_1/\alpha_1$ and $s_1^i = \alpha_1^i s_1/\alpha_1$. Note that $y_1^i + s_1^i = \alpha_1^i y_1/\alpha_1 + \alpha_1^i s_1/\alpha_1 = \alpha_1^i (y_1 + s_1)/\alpha_1 = \alpha_1^i$. Furthermore, $\sum_{i=1}^k y_1^i = \sum_{i=1}^k \alpha_1^i y_1/\alpha_1 = (y_1/\alpha_1) \sum_{i=1}^k \alpha_1^i = (y_1/\alpha_1)\alpha_1 = y_1$ and $\sum_{i=1}^k s_1^i = \sum_{i=1}^k \alpha_1^i s_1/\alpha_1 = (s_1/\alpha_1) \sum_{i=1}^k \alpha_1^i = (s_1/\alpha_1)\alpha_1 = s_1$.

Inductive Hypothesis: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^{n-1}$ and $y \in \Gamma(K, \alpha)$. For $i \in \{1, ..., k\}$, we are given $\alpha^i \in \{0, R^+\}^{n-1}$ such that $\sum_{i=1}^k \alpha^i = \alpha$ then we can find $y^i \in \Gamma(K, \alpha^i)$ such that $\sum_{i=1}^k y^i = y$ and $\sum_{i=1}^k s^i = s$ where s is the vector of slack variables corresponding to y and s^i is the vector of slack variables corresponding to y^i.

We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$ and $y \in \Gamma(K, \alpha)$. Let s be the vector of slack variables corresponding to y. We will assume that $\alpha_n > 0$. If $\alpha_n = 0$ then $y_n = 0$ and $s_n = 0$ and the result follows trivially from the induction hypothesis. There are two cases to consider.

Case 1 : $2 \leq n \leq m$. Since $y \in \Gamma(K, \alpha)$, by Lemma 5.1,

$$\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} = \sum_{i=1}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}$$

We let $y_n^j = \alpha_n^j y_n/\alpha_n$ and $s_n^j = \alpha_n^j s_n/\alpha_n$ (Note that $\sum_{j=1}^k y_n^j = \sum_{j=1}^k \alpha_n^j y_n/\alpha_n = (y_n/\alpha_n) \sum_{j=1}^k \alpha_n^j = (y_n/\alpha_n)\alpha_n = y_n$ and $\sum_{j=1}^k s_n^j = \sum_{j=1}^k \alpha_n^j s_n/\alpha_n = (s_n/\alpha_n) \sum_{j=1}^k \alpha_n^j = (s_n/\alpha_n)\alpha_n = s_n$). Next note that for $i \in \{1, ..., n-1\}$ $\sum_{j=1}^k \alpha_i^j + (\alpha_n^j - y_n^j) K_{n-i} = \sum_{j=1}^k \alpha_i^j + K_{n-i} \sum_{j=1}^k \alpha_n^j - K_{n-i} \sum_{j=1}^k y_n^j = \alpha_i + (\alpha_n - y_n) K_{n-i}$. By the inductive hypothesis we can find $\bar{y}_1, ..., \bar{y}_{n-1}$and $\bar{s}_1, ..., \bar{s}_{n-1}$ such that for $i \in \{1, ..., n-1\}$ $\sum_{j=1}^k y_i^j = y_i$ and $\sum_{j=1}^k s_i^j = s_i$ and,

$$\sum_{i=1}^{n-1} y_i^j A_i^{n-1} + \sum_{i=1}^{n-1} s_i^j I_i^{n-1} = \sum_{i=1}^{n-1} [\alpha_i^j + (\alpha_n^j - y_n^j) K_{n-i}] A_i^{n-1}$$

$$\begin{aligned}
&\sum_{i=1}^{n-1} y_i^j A_i^n + \sum_{i=1}^{n-1} s_i^j I_i^n = \sum_{i=1}^{n-1} [\alpha_i^j + (\alpha_n^j - y_n^j) K_{n-i}] A_i^n \\
&= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + \sum_{i=1}^{n-1} [(\alpha_n^j - y_n^j) K_i] A_{n-i}^n \\
&= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + (\alpha_n^j - y_n^j)(A_n^n - I_n^n)
\end{aligned}$$

Hence,

$$\begin{aligned}
&\sum_{i=1}^{n} y_i^j A_i^n + \sum_{i=1}^{n} s_i^j I_i^n \\
&= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + (\alpha_n^j - y_n^j)(A_n^n - I_n^n) + y_n^j A_n^n + s_n^j I_n^n \\
&= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + s_n^j (A_n^n - I_n^n) + y_n^j A_n^n + s_n^j I_n^n \\
&= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + s_n^j A_n^n + y_n^j A_n^n \\
&= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + \alpha_n^j A_n^n = \sum_{i=1}^{n} \alpha_i^j A_i^n
\end{aligned}$$

Case 2 : $n > m$. Since $y \in \Gamma(K, \alpha)$, by Lemma 5.1,

$$\begin{aligned}
&\sum_{i=1}^{n-1} y_i A_i^{n-1} + \sum_{i=1}^{n-1} s_i I_i^{n-1} \\
&= \sum_{i=1}^{n-m-1} \alpha_i A_i^{n-1} + \sum_{i=n-m}^{n-1} [\alpha_i + (\alpha_n - y_n) K_{n-i}] A_i^{n-1}
\end{aligned}$$

We let $y_n^j = \alpha_n^j y_n/\alpha_n$ and $s_n^j = \alpha_n^j s_n/\alpha_n$ (Note that $\sum_{j=1}^k y_n^j = \sum_{j=1}^k \alpha_n^j y_n/\alpha_n = (y_n/\alpha_n) \sum_{j=1}^k \alpha_n^j = (y_n/\alpha_n)\alpha_n = y_n$ and $\sum_{j=1}^k s_n^j = \sum_{j=1}^k \alpha_n^j s_n/\alpha_n = (s_n/\alpha_n) \sum_{j=1}^k \alpha_n^j = (s_n/\alpha_n)\alpha_n = s_n$). Next note that for $i \in \{1, ..., n-m-1\}$ $\sum_{j=1}^k \alpha_i^j = \alpha_i$. And for $i \in \{n-m, ..., n\}$ $\sum_{j=1}^k [\alpha_i^j + (\alpha_n^j - y_n^j) K_{n-i}] = \sum_{j=1}^k \alpha_i^j + K_{n-i} \sum_{j=1}^k \alpha_n^j - K_{n-i} \sum_{j=1}^k y_n^j = \sum_{j=1}^k \alpha_i^j + (\sum_{j=1}^k \alpha_n^j - \sum_{j=1}^k y_n^j) K_{n-i} = \alpha_i + (\alpha_n - y_n) K_{n-i}$. By the inductive hypothesis we can find $\bar{y}_1, ..., \bar{y}_{n-1}$and $\bar{s}_1, ..., \bar{s}_{n-1}$ such that

for $i \in \{1,...,n-1\}$ $\sum_{j=1}^{k} y_i^j = y_i$ and $\sum_{j=1}^{k} s_i^j = s_i$ and,

$$\sum_{i=1}^{n-1} y_i^j A_i^{n-1} + \sum_{i=1}^{n-1} s_i^j I_i^{n-1}$$
$$= \sum_{i=1}^{n-m-1} \alpha_i^j A_i^{n-1} + \sum_{i=n-m}^{n-1} [\alpha_i^j + (\alpha_n^j - y_n^j) K_{n-i}] A_i^{n-1}$$

$$\sum_{i=1}^{n-1} y_i^j A_i^n + \sum_{i=1}^{n-1} s_i^j I_i^n$$
$$= \sum_{i=1}^{n-m-1} \alpha_i^j A_i^n + \sum_{i=n-m}^{n-1} [\alpha_i^j + (\alpha_n^j - y_n^j) K_{n-i}] A_i^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + \sum_{i=n-m}^{n-1} (\alpha_n^j - y_n^j) K_{n-i} A_i^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + \sum_{i=1}^{m} [(\alpha_n^j - y_n^j) K_i] A_{n-i}^n$$

Hence,

$$\sum_{i=1}^{n} y_i^j A_i^n + \sum_{i=1}^{n} s_i^j I_i^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + \sum_{i=1}^{m} [(\alpha_n^j - y_n^j) K_i] A_{n-i}^n + y_n^j A_n^n + s_n^j I_n^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + (\alpha_n^j - y_n^j) \sum_{i=1}^{m} K_i A_{n-i}^n + y_n^j A_n^n + s_n^j I_n^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + s_n^j (A_n^n - I_n^n) + y_n^j A_n^n + s_n^j I_n^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + s_n^j A_n^n + y_n^j A_n^n$$
$$= \sum_{i=1}^{n-1} \alpha_i^j A_i^n + \alpha_n^j A_n^n = \sum_{i=1}^{n} \alpha_i^j A_i^n$$ ■

Theorem 5.2: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$ and $c \in R^n$. Let y_{opt} maximize cy subject to $y \in \Gamma(K, \alpha)$ and let C be the maximum of cy subject to $y \in \Gamma(K, \alpha)$. Let y_{opt}^i maximize cy subject to $y \in \Gamma(K, I_i^n)$ and let C^i be the maximum of cy subject to $y \in \Gamma(K, I_i^n)$. Equivalently, let C^i be the maximum of $[c_1, ..., c_i]y$ subject to $y \in \Gamma(K, I_i^i)$. We claim that $C = \sum_{i=1}^{n} \alpha_i C^i$.

Proof: Let y_{opt} maximize cy subject to $y \in \Gamma(K, \alpha)$ and let s_{opt} be the slack variables corresponding to y_{opt}. For all $i \in \{1,...,n\}$ let x_{opt}^i maximize cx subject to $x \in \Gamma(K, \alpha_i I_i^n)$ and let r_{opt}^i be the slack variables corresponding to x_{opt}^i. We know that $\alpha = \sum_{i=1}^{n} \alpha_i I_i^n$. Hence, by Lemma 5.3 we can find $x^i \in \Gamma(K, \alpha_i I_i^n)$ such that $\sum_{i=1}^{n} x^i = y_{opt}$ and $\sum_{i=1}^{n} r^i = s_{opt}$. We will next show that $cx^i = cx_{opt}^i$. Let $C = cy_{opt}$. Since $\sum_{i=1}^{n} x^i = y$, $C = c\sum_{i=1}^{n} x^i = \sum_{i=1}^{n} cx^i$. Assume that there exists an i such that $cx_{opt}^i > cx^i$. Let $\bar{y} = (x_{opt}^i - x^i) + \sum_{j=1}^{n} x^j$ and let $\bar{s} = (r_{opt}^i - r^i) + \sum_{j=1}^{n} r^j$. Next we show that $\bar{y} \in \Gamma(K, \alpha)$. First note that both $\bar{y}, \bar{s} \in \{0, R^+\}^n$. Also,

$$A^n \bar{y} + I^n \bar{s} = A^n[(x_{opt}^i - x^i) + \sum_{j=1}^{n} x^j] + I^n[(r_{opt}^i - r^i) + \sum_{j=1}^{n} r^j]$$
$$= (A^n x_{opt}^i + I^n r_{opt}^i) - (A^n x^i + I^n r^i) + (A^n y_{opt} + I^n s_{opt})$$

Since $x_{opt}^i \in \Gamma(K, \alpha_i I_i^n)$, $A^n x_{opt}^i + I^n r_{opt}^i = A^n \alpha_i I_i^n$. Similarly, $A^n x^i + I^n r^i = A^n \alpha_i I_i^n$ and $A^n y_{opt} + I^n s_{opt} = A^n \alpha$. Hence,

$$A^n \bar{y} + I^n \bar{s} = A^n \alpha_i I_i^n - A^n \alpha_i I_i^n + A^n \alpha = A^n \alpha$$

Thus $\bar{y} \in \Gamma(K, \alpha)$. Now note that $c\bar{y} = (cx_{opt}^i - cx^i) + \sum_{j=1}^{n} cx^j = (cx_{opt}^i - cx^i) + C > C$. Hence, y_{opt} does not maximize cy subject to $y \in \Gamma(K, \alpha)$. This is a contradiction. Hence, for all $i \in \{1,...,n\}$ $cx^i = cx_{opt}^i$. Thus, $C = cy_{opt} = \sum_{i=1}^{n} cx^i = \sum_{i=1}^{n} cx_{opt}^i$. It is easy to show that $cx_{opt}^i = \alpha_i cy_{opt}^i$ where y_{opt}^i maximizes cy subject to $y \in \Gamma(K, I_i^n)$. Hence, $C = \sum_{i=1}^{n} cx_{opt}^i = \sum_{i=1}^{n} \alpha_i cy_{opt}^i = \sum_{i=1}^{n} \alpha_i C^i$, where $C^i = cy_{opt}^i$. ■

VI. Algorithms

The 0-1 Principle and the Decomposition Principle lead directly to efficient algorithms for maximizing linear functions over Generalized Fibonacci Polyhedra and minimizing linear functions over the intersection of a Generalized Fibonacci Polyhedra and a hyperplane.

A. Maximization of Linear Functions over Generalized Fibonacci Polyhedra

In this subsection we will state an algorithm for maximizing an arbitrary linear function over a Generalized Fibonacci Polyhedra. The algorithm we present is linear in the dimension of the polyhedra and is a consequence of the 0-1 Principle and the Decomposition Principle. Before we state the algorithm we will state the problem formally.

Problem 6.1: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$ and $c \in R^n$. Find $C = \max cy$ subject to $y \in \Gamma(K, \alpha)$.

Algorithm 6.1: For $i \in \{1,....,n\}$ we compute C^i using the following recurrence,

$$C^i = \begin{cases} \max(\sum_{j=1}^{m} K_j C^{i-j}, c_i) & m < i \leq n \\ \max(\sum_{j=1}^{i-1} K_j C^{i-j}, c_i) & 2 \leq i \leq m \\ \max(0, c_1) & i = 1 \end{cases}$$

Note that this can be accomplished in $O(nm)$ operations by first computing C^1, followed by C^2, followed by C^3, and so on and so forth. Now we let $C = \sum_{i=1}^{n} \alpha_i C^i$. This computation requires only $O(n)$ operations.

Proof: If we can show that for $i \in \{1,....,n\}$ C^i is the maximum of cy subject to $y \in \Gamma(K, I_i^i)$ then from the Decomposition Principle it will follow that $C = \sum_{i=1}^{n} \alpha_i C^i$. To complete the proof it would suffice to establish that this is the case. Let y^i maximize cy subject to $y \in \Gamma(K, I_i^i)$ and let s^i be the corresponding vector of slack variables. Since y^i maximizes cy subject to $y \in \Gamma(K, I_i^i)$, y^i must

be an extreme point of $\Gamma(K, I_i^i)$. There are three cases to consider.

Case 1 : $i = 1$. From the 0-1 Principle we know that $(y_1^1, s_1^1) = (1,0)$ or $(y_1^1, s_1^1) = (0,1)$. If $(y_1^1, s_1^1) = (1,0)$ then cy^1 is c_1. If $(y_1^1, s_1^1) = (0,1)$ then cy^1 is 0. Hence $C^1 = \max(0, c_1)$.

Case 2 : $2 \leq i \leq m$. From the 0-1 Principle, it follows that $(y_i^i, s_i^i) = (0,1)$ or $(y_i^i, s_i^i) = (1,0)$. If $(y_i^i, s_i^i) = (1,0)$, it is easy to show that for $j \in \{1, ..., i-1\}$, $y_j^i = 0$ and $s_j^i = 0$. Hence, $cy^i = c_i$. Now consider the case when $(y_i^i, s_i^i) = (0,1)$. We know that,

$$\sum_{j=1}^{i} y_j^i A_j^i + \sum_{j=1}^{i} s_j^i I_j^i = I_i^i + \sum_{j=1}^{i-1} y_j^i A_j^i + \sum_{j=1}^{i-1} s_j^i I_j^i = A_i^i$$

Hence,

$$\sum_{j=1}^{i-1} y_j^i A_j^i + \sum_{j=1}^{i-1} s_j^i I_j^i = A_i^i - I_i^i = I_i^i + \sum_{j=1}^{i-1} K_j A_{i-j}^i - I_i^i$$

Hence,

$$\sum_{j=1}^{i-1} y_j^i A_j^i + \sum_{j=1}^{i-1} s_j^i I_j^i = \sum_{j=1}^{i-1} K_j A_{i-j}^i$$

We let $\bar{c} = (c_1, ..., c_{i-1})$ and $\bar{y} = (y_1^i, ..., y_{i-1}^i)$. Also we define $\bar{\alpha} = (K_{i-1}, ..., K_1)$ and $\bar{C}$ to be the maximum of $\bar{c}\bar{y}$ subject to $\bar{y} \in \Gamma(K, \bar{\alpha})$. It is easy to show that the maximum of cy subject to $y \in \Gamma(K, I_i^i)$ and $(y_i, s_i) = (0,1)$ is just $\bar{C}$. From the Decomposition Principle we know that $\bar{C} = \sum_{j=1}^{i-1} \bar{\alpha}_j C^j = \sum_{j=1}^{i-1} K_{i-j} C^j = \sum_{j=1}^{i-1} K_j C^{i-j}$. We have shown that the maximum of cy subject to $y \in \Gamma(K, I_i^i)$ and $(y_i, s_i) = (1,0)$ is c_i and the maximum of cy subject to $y \in \Gamma(K, I_i^i)$ and $(y_i, s_i) = (0,1)$ is $\sum_{j=1}^{i-1} K_j C^{i-j}$. Hence $C^i = \max(\sum_{j=1}^{i-1} K_j C^{i-j}, c_i)$.

Case 3 : $i > m$. This case is similar to Case 2. We will simply state that in this case $C^i = \max(\sum_{j=1}^{m} K_j C^{i-j}, c_i)$ and will leave the proof as an exercise for the reader. ■

B. Minimization of Linear Functions over the Intersection of a Generalized Fibonacci Polyhedra and a Hyperplane

In this subsection we will state an algorithm for minimizing an arbitrary linear function over the intersection of a Generalized Fibonacci Polyhedra and a hyperplane. The algorithm we present is polynomial in the dimension of the polyhedra and is a consequence of the 0-1 Principle and the Decomposition Principle. Before we state the algorithm we will state the problem formally.

Problem 6.2: We are given $K \in \{0,1\}^m$, $\alpha \in \{0, R^+\}^n$, $c \in R^n$ and $p \in R^n$ and $k \in R$. Find $C = \min cy$ subject to $py = k$ and $y \in \Gamma(K, \alpha)$. Equivalently, find $C = \min cy$ subject to $py = k$, $A^n y \leq A^n \alpha$ and $y \in \{0, R^+\}^n$ where A^n is an $n \times n$ matrix such that $y \in \Gamma(K, \alpha)$ iff $A^n y \leq A^n \alpha$ and $y \in \{0, R^+\}^n$.

Let π be the dual variable associated with the constraint $py = k$ and let μ be the vector of dual variables associated with the constraints $A^n y \leq A^n \alpha$. The dual of the linear program above is

$$\begin{array}{rc} \max & \pi k + \mu A^n \alpha \\ & \pi p + \mu A^n \leq c \\ \text{subject to} & \pi \text{ unrestricted}, \\ & \mu \in \{0, R^-\}^n \end{array}$$

We next decompose the above problem as follows

$$\max_{\pi \text{ unrestricted}} \left\{ \pi k + \begin{array}{cc} \max & \mu A^n \alpha \\ \text{subject to} & \mu A^n \leq c - \pi p \\ & \mu \in \{0, R^-\}^n \end{array} \right\}$$

Let $\bar{y}$ be the set of dual variables associated with the constraint $\mu A^n \leq c - \pi p$. We replace the inner linear program by its dual (instead of minimizing, we maximize the negative of the cost function)

$$\max_{\pi \text{ unrestricted}} \left\{ \pi k - \begin{array}{cc} \max & (\pi p - c)\bar{y} \\ \text{subject to} & A^n \bar{y} \leq A^n \alpha \\ & \bar{y} \in \{0, R^+\}^n \end{array} \right\}$$

Next we define $f(\pi) = \pi k$, $g(\pi) = \max\ (\pi p - c)\bar{y}$ subject to $A^n \bar{y} \leq A^n \alpha$, $\bar{y} \in \{0, R^+\}^n$ and $L(\pi) = f(\pi) - g(\pi)$.Using the results in the previous section, we know that for any given value of π, $g(\pi) = \sum_{i=1}^{n} \alpha_i g_i(\pi)$ where

$$g_i(\pi) = \begin{cases} \max(\sum_{j=1}^{m} K_j g_{i-j}(\pi), \pi p_i - c_i) & m < i \leq n \\ \max(\sum_{j=1}^{i-1} K_j g_{i-j}(\pi), \pi p_i - c_i) & 2 \leq i \leq m \\ \max(0, \pi p_1 - c_1) & i = 1 \end{cases}$$

We know that $g(\pi) = L(\pi) - f(\pi)$ where $f(\pi) = \pi k$. f is thus linear and continuous and L is the Lagrangian Dual of the linear program and is thus piecewise linear, concave and continuous. Therefore, g must be piecewise linear and convex. Using a similar argument it is possible to show that g_i will be piecewise linear, convex and continuous. Piecewise linear and continuous functions can be represented by a sequence of breakpoints (stored in sorted order) which partition the real axis into nonoverlapping intervals. Within each interval the piecewise linear function is linear and this line can be represented using two parameters describing the slope and the intercept. Addition of two piecewise linear functions results in a piecewise linear function whose parametric representation can be computed in linear time (linear in number of breakpoints). Similarly the maximum of a linear function and a piecewise linear function is a piecewise linear function whose parametric representation can be computed in linear time (linear in the number of breakpoints). Our proposal is to compute the parametric representation of g_1, g_2, ... , g_n and then compute the parametric representation of g and finally L.

Given the parametric representation of L, it is easy to determine the value of π that maximizes L and thus the maximum value of L. For this process to be efficient we need to bound the number of breakpoints of g_1, g_2, ... , g_n and g and L. We do this next.

Let $b_i = \{\pi | \pi$ is a breakpoint in $g_i\}$ and let $B_i = \{\pi | \pi$ is a breakpoint in g_1 or g_2 or .... or $g_i\}$. Note that $b_i \subseteq B_i$ and hence $|b_i| \leq |B_i|$. We claim that $|B_i| \leq 2i-1$. The proof is by induction. $|B_1|$ can be atmost one since g_1 can have at most one breakpoint. Next we assume that $|B_{i-1}|$ can be atmost $2(i-1)-1 = 2i-3$. First consider the case when $2 \leq i \leq m$. $\sum_{j=1}^{i-1} K_j g_{i-j}$ will be a piecewise linear and convex. Furthermore, if π is a breakpoint of $\sum_{j=1}^{i-1} K_j g_{i-j}$ then π is a breakpoint of g_1 or π is a breakpoint of g_2 or π is a breakpoint of g_3 or .... or π is a breakpoint of g_{i-1}. Hence the number of breakpoints of $\sum_{j=1}^{i-1} K_j g_{i-j}$ is less than or equal to $|B_{i-1}|$. Note that $\sum_{j=1}^{i-1} K_j g_{i-j}$ is a piecewise linear, convex and continuous function and $\pi p_i - c_i$ is a straight line. A straight line can "intersect" a piecewise linear, convex and continuous function at atmost two points (if the line overlaps a line segment, then the number of breakpoints is unaltered). Thus, $|B_i| \leq |B_{i-1}| + 2 = 2i-3+2 = 2i-1$. The proof for the case when $i > m$ is similar to the previous case and left as an exercise for the reader. Now note that $g(\pi) = \sum_{i=1}^{n} \alpha_i g_i(\pi)$. π is a breakpoint of g implies π is a breakpoint of g_1 or π is a breakpoint of g_2 or π is a breakpoint of g_3 or ... or π is a breakpoint of g_n. Hence, the number of breakpoints of g is less than or equal to $|B_n|$ which is equal to $2n-1$. Since g has atmost $2n-1$ breakpoints and f is linear, L has atmost $2n-1$ breakpoints.

VII. Applications and Examples

We will illustrate how the techniques developed in the previous section can be used to determine if the linear programming relaxation of the integer program has a solution, and find the optimal solution if one exists.

Example 7.1: We are given $S = \{1,2,3\}$, $T_E = 7$, $T_D = 8$ and $R = 6/7 = .857$. We would like to determine if the linear programming relaxation of the integer program has a solution, and find an optimal solution if one exists. The linear programming relaxation of the integer program is given below.

$$\min \sum_{i=1}^{T_D} \sum_{j=1}^{T_E} x_j^i \text{ , subject to}$$
$$\sum_{i=1}^{T_E} 2^{(T_E-i)} x_i = 2^{T_E}$$
$$1 \leq L \leq T_D, x^L + \sum_{l=1}^{L-1} N_K(L-l) x^l \leq N_K(L)$$
$$j/i < R \Longrightarrow x_j^i = 0,\ x_j^i \geq 0$$

Notice that without loss of generality we can assume that the only non zero variables are x_1^1, x_2^2, x_3^3, x_4^4, x_5^5, x_6^6, x_6^7 and x_7^8. Consider an example – we claim that we can assume $x_3^2 = 0$. Assume $x_3^2 > 0$. We increase x_2^2 by $x_3^2/2$ and set $x_3^2 = 0$. Notice that x^2 decreased and thus the decoder inequalities will still be satisfied. However, x_3 decreased by x_3^2 and x_2 increased by $x_3^2/2$. Since the coefficient associated with x_3 is 16 and the coefficient associated with x_2 is 32 the encoder equality will still be satisfied. Also the value of the cost function has decreased. Thus, in an optimal solution $x_3^2 = 0$. The linear program above can be reduced to the linear program below.

$$\begin{array}{ll} \min & cx \\ \text{subject to} & px = k,\ A^n x \leq A^n \alpha,\ x \in \{0, R^+\}^n \end{array}$$

where $c = [1,1,1,1,1,1,1,1]$, $x = [x_7^8, x_6^7, x_6^6, x_5^5, x_4^4, x_3^3, x_2^2, x_1^1]^T$, $p = [1,2,2,4,8,16,32,64]$, $k = 128$, $\alpha = [0,0,0,0,0,1,1,1]$ and

$$A^n = \begin{bmatrix} 1 & 1 & 2 & 4 & 7 & 13 & 24 & 44 \\ 0 & 1 & 1 & 2 & 4 & 7 & 13 & 24 \\ 0 & 0 & 1 & 1 & 2 & 4 & 7 & 13 \\ 0 & 0 & 0 & 1 & 1 & 2 & 4 & 7 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

A sufficient and necessary condition for the linear programming relaxation of the integer program to have a solution is that the maximum attained by the linear program below be greater than or equal to k.

$$\begin{array}{ll} \max & px \\ \text{subject to} & A^n x \leq A^n \alpha,\ x \in \{0, R^+\}^n \end{array}$$

This is a linear programming problem over a Generalized Fibonacci Polyhedra and can be solved efficiently using the algorithm developed in this paper. We solve the problem below.

$$C^1 = \max(0, c_1) = \max(0,1) = 1$$
$$C^2 = \max(\textstyle\sum_{j=1}^{1} K_j C_{2-j}, c_2) = \max(1,2) = 2$$
$$C^3 = \max(\textstyle\sum_{j=1}^{2} K_j C_{3-j}, c_3) = \max(1+2,2) = 3$$
$$C^4 = \max(\textstyle\sum_{j=1}^{3} K_j C_{4-j}, c_4) = \max(1+2+3,4) = 6$$
$$C^5 = \max(\textstyle\sum_{j=1}^{3} K_j C_{5-j}, c_5) = \max(2+3+6,8) = 11$$
$$C^6 = \max(\textstyle\sum_{j=1}^{3} K_j C_{6-j}, c_6) = \max(3+6+11,16) = 20$$
$$C^7 = \max(\textstyle\sum_{j=1}^{3} K_j C_{7-j}, c_7) = \max(6+11+20,32) = 37$$
$$C^8 = \max(\textstyle\sum_{j=1}^{3} K_j C_{8-j}, c_8)$$
$$= \max(11+20+37,64) = 68$$

$$C = C^6 + C^7 + C^8 = 20 + 37 + 68 = 125$$

Note that the maximum that is attained is just 125 which falls short of the desired 128. This means that the linear

programming relaxation of the integer program does not have a solution.

Example 7.2: We are given $S = \{1, 2, 3\}$, $T_E = 5$, $T_D = 6$ and $R = 4/5 = 0.80$. We would like to determine if the linear programming relaxation of the integer program has a solution, and find an optimal solution if one exists. The linear programming relaxation of the integer program is given below.

$$\begin{array}{l} \min \quad \sum_{i=1}^{T_D} \sum_{j=1}^{T_E} x_j^i \ \text{, subject to} \\ \sum_{i=1}^{T_E} 2^{(T_E - i)} x_i = 2^{T_E} \\ 1 \leq L \leq T_D, x^L + \sum_{l=1}^{L-1} N_K(L-l) x^l \leq N_K(L) \\ j/i < R \Longrightarrow x_j^i = 0, \ x_j^i \geq 0 \end{array}$$

Notice that without loss of generality we can assume that the only non zero variables are x_1^1, x_2^2, x_3^3, x_4^4, x_4^5 and x_5^6. Consider an example – we claim that we can assume $x_3^2 = 0$. Assume $x_3^2 > 0$. We increase x_2^2 by $x_3^2/2$ and set $x_3^2 = 0$. Notice that x^2 decreased and thus the decoder inequalities will still be satisfied. However, x_3 decreased by x_3^2 and x_2 increased by $x_3^2/2$. Since the coefficient associated with x_3 is 4 and the coefficient associated with x_2 is 8, the encoder equality will still be satisfied. Also the value of the cost function has decreased. Thus, in anll optimal solution $x_3^2 = 0$. The linear program above can be reduced to the linear program below.

$$\begin{array}{ll} \min & cx \\ \text{subject to} & px = k, \ A^n x \leq A^n \alpha, \ x \in \{0, R^+\}^n \end{array}$$

where $c = [1, 1, 1, 1, 1, 1]$, $x = [x_5^6, x_4^5, x_4^4, x_3^3, x_2^2, x_1^1]^T$, $p = [1, 2, 2, 4, 8, 16]$, $k = 32$, $\alpha = [0, 0, 0, 1, 1, 1]$ and

$$A^n = \begin{bmatrix} 1 & 1 & 2 & 4 & 7 & 13 \\ 0 & 1 & 1 & 2 & 4 & 7 \\ 0 & 0 & 1 & 1 & 2 & 4 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

A sufficient and necessary condition for the linear programming relaxation of the integer program to have a solution is that the maximum attained by the linear program below be greater than or equal to k.

$$\begin{array}{ll} \max & px \\ \text{subject to} & A^n x \leq A^n \alpha, \ x \in \{0, R^+\}^n \end{array}$$

This is a linear programming problem over a Generalized Fibonacci Polyhedra and can be solved efficiently using the algorithm developed in this paper. We solve the problem below.

$$\begin{array}{l} C^1 = \max(0, p_1) = \max(0, 1) = 1 \\ C^2 = \max(\sum_{j=1}^{1} K_j C_{2-j}, p_2) = \max(1, 2) = 2 \\ C^3 = \max(\sum_{j=1}^{2} K_j C_{3-j}, p_3) = \max(1+2, 2) = 3 \\ C^4 = \max(\sum_{j=1}^{3} K_j C_{4-j}, p_4) = \max(1+2+3, 4) = 6 \\ C^5 = \max(\sum_{j=1}^{3} K_j C_{5-j}, p_5) = \max(2+3+6, 8) = 11 \\ C^6 = \max(\sum_{j=1}^{3} K_j C_{6-j}, p_6) = \max(3+6+11, 16) = 20 \end{array}$$

$$C = C^4 + C^5 + C^6 = 6 + 11 + 20 = 37$$

Note that the maximum that is attained is 37 which exceeds the desired 32. This means that the linear programming relaxation of the integer program has a solution. We next illustrate how to compute the optimal solution to the linear programming relaxation of the integer program. The linear programming relaxation of the integer program is given below.

$$\begin{array}{ll} \min & cx \\ \text{subject to} & px = k, \ A^n x \leq A^n \alpha, \ x \in \{0, R^+\}^n \end{array}$$

This is equivalent to maximizing $L(\pi) = f(\pi) - g(\pi)$ where π is unrestricted, $f(\pi) = \pi k$ and

$$g(\pi) = \left\{ \begin{array}{ll} \max & (\pi p - c)\bar{x} \\ \text{subject to} & A^n \bar{x} \leq A^n \alpha, \ \bar{x} \in \{0, R^+\}^n \end{array} \right\}$$

Next, we compute $g_1, g_2, ..., g_5, g_6$.

$$g_1(\pi) = \max \left\{ \begin{array}{l} 0 \\ p_1\pi - c_1 \end{array} \right. = \left\{ \begin{array}{ll} 0 & \pi \leq 1 \\ \pi - 1 & \pi > 1 \end{array} \right.$$

$$g_2(\pi) = \max \left\{ \begin{array}{l} \sum_{j=1}^{1} K_j g_{2-j}(\pi) \\ p_2\pi - c_2 \end{array} \right. = \left\{ \begin{array}{ll} 0 & \pi \leq \frac{1}{2} \\ 2\pi - 1 & \pi > \frac{1}{2} \end{array} \right.$$

$$g_3(\pi) = \max \left\{ \begin{array}{l} \sum_{j=1}^{2} K_j g_{3-j}(\pi) \\ p_3\pi - c_3 \end{array} \right. = \left\{ \begin{array}{ll} 0 & \pi \leq \frac{1}{2} \\ 2\pi - 1 & \frac{1}{2} < \pi \leq 1 \\ 3\pi - 2 & 1 < \pi \end{array} \right.$$

$$g_4(\pi) = \max \left\{ \begin{array}{l} \sum_{j=1}^{3} K_j g_{4-j}(\pi) \\ p_4\pi - c_4 \end{array} \right. = \left\{ \begin{array}{ll} 0 & \pi \leq \frac{1}{4} \\ 4\pi - 1 & \frac{1}{4} < \pi \leq \frac{3}{2} \\ 6\pi - 4 & \frac{3}{2} < \pi \end{array} \right.$$

$$g_5(\pi) = \max \left\{ \begin{array}{l} \sum_{j=1}^{3} K_j g_{5-j}(\pi) \\ p_5\pi - c_5 \end{array} \right. = \left\{ \begin{array}{ll} 0 & \pi \leq \frac{1}{8} \\ 8\pi - 1 & \frac{1}{8} < \pi \leq 2 \\ 11\pi - 7 & 2 < \pi \end{array} \right.$$

$$g_6(\pi) = \max \left\{ \begin{array}{c} \sum_{j=1}^{3} K_j g_{6-j}(\pi) \\ p_6\pi - c_6 \end{array} \right. = \left\{ \begin{array}{ll} 0 & \pi \leq \frac{1}{16} \\ 16\pi - 1 & \frac{1}{16} < \pi \leq 3 \\ 20\pi - 13 & 3 < \pi \end{array} \right.$$

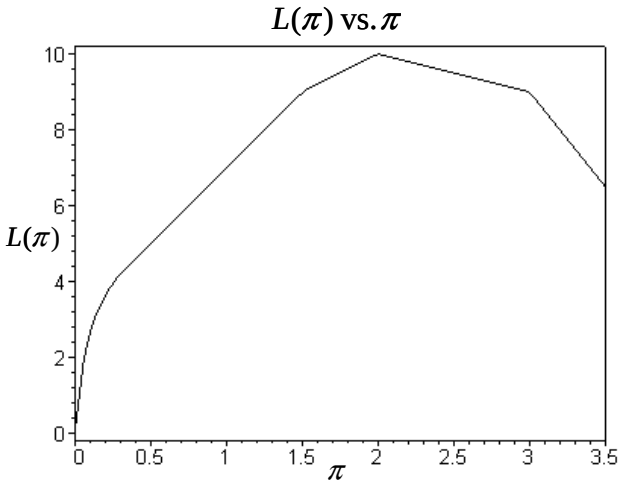
Hence,

$$g(\pi) = g_4(\pi)+g_5(\pi)+g_6(\pi) = \left\{ \begin{array}{ll} 0 & \pi \leq \frac{1}{16} \\ 16\pi - 1 & \frac{1}{16} < \pi \leq \frac{1}{8} \\ 24\pi - 2 & \frac{1}{8} < \pi \leq \frac{1}{4} \\ 28\pi - 3 & \frac{1}{4} < \pi \leq \frac{3}{2} \\ 30\pi - 6 & \frac{3}{2} < \pi \leq 2 \\ 33\pi - 12 & 2 < \pi \leq 3 \\ 37\pi - 24 & \pi > 3 \end{array} \right.$$

Hence,

$$L(\pi) = f(\pi) - g(\pi) = \left\{ \begin{array}{ll} 32\pi & \pi \leq \frac{1}{16} \\ 16\pi + 1 & \frac{1}{16} < \pi \leq \frac{1}{8} \\ 8\pi + 2 & \frac{1}{8} < \pi \leq \frac{1}{4} \\ 4\pi + 3 & \frac{1}{4} < \pi \leq \frac{3}{2} \\ 2\pi + 6 & \frac{3}{2} < \pi \leq 2 \\ -\pi + 12 & 2 < \pi \leq 3 \\ -5\pi + 24 & \pi > 3 \end{array} \right.$$

The maximum value taken on by $L(\pi)$ is 10 and occurs at $\pi = 2$. $L(\pi)$ is plotted below.



We next turn our attention to the integer program associated with this code construction problem. Using a standard ILP solver we find that the solution to the integer linear program is $x_1^1 = 1$, $x_4^5 = 6$ and $x_5^6 = 4$ (all other variables are 0). Hence the size of the code will be 11. Thus the solution to the linear program provides a lower bound on the size of the code. Next we illustrate how to construct the code itself.

First we contruct a prefix-free set using the source alphabet $\{0, 1\}$ such that the number of elements consisting of 1 bit is 1, the number of elements consisting of 4 bits is 6 and the number of elements consisting of 5 bits is 4. A valid set is $\{0, 1000, 1001, 1010, 1011, 1100, 1101, 11100, 11101, 11110, 11111\}$. We will call this set A. Next we construct a prefix-free set using the code alphabet $\{1, 2, 3\}$ such that the number of elements having a duration of 1 unit is 1, the number of elements having a duration of 5 units is 6 and the number of elements having a duration of 6 units is 4. A valid set is $\{1, 2111, 212, 221, 23, 311, 32, 2112, 213, 312, 33\}$. We will call this set B. Now we arbitrarily map 1 element consisting of 1 bit from set A to 1 element of duration 1 in set B. We arbitrarily map 6 elements consisting of 4 bits in set A to 6 elements of duration 5 in set B in a one on one manner. Similarly, we arbitrarily map 4 elements consisting of 5 bits in set A to 4 elements of duration 6 in set B in a one on one manner. A valid mapping (code) is shown in the table below.

$0 \leftrightarrow 1$	$1011 \leftrightarrow 23$	$11101 \leftrightarrow 213$
$1000 \leftrightarrow 2111$	$1100 \leftrightarrow 311$	$11110 \leftrightarrow 312$
$1001 \leftrightarrow 212$	$1101 \leftrightarrow 32$	$11111 \leftrightarrow 33$
$1010 \leftrightarrow 221$	$11100 \leftrightarrow 2112$	

VIII. Generalized Fibonacci Numbers and their Sums

Two series that arise in both the context of rate analysis and code construction are the Generalized Fibonacci Numbers and their sums. Given a set of symbols S, the Tth Generalized Fibonacci Number is $N_K(T)$ and the sum of the first T Generalized Fibonacci Numbers is denoted by $\bar{N}_K(T) = \sum_{t=1}^{T} N_K(t)$. In this section, we will first derive a closed form expression for the Generalized Fibonacci Numbers. This derivation is similar to that of Capocelli and Cull [9] and Mukhtar and Bruck [37].

Theorem 8.1: Let $p(\lambda) = \lambda^m - \sum_{i=1}^{m} K_i \lambda^{(m-i)}$ and $p'(\lambda) = m\lambda^{m-1} - \sum_{i=1}^{m} K_i (m-i)\lambda^{(m-i-1)}$. For $i \in \{1, 2, ..., m-1, m\}$ ϕ_i be the roots of $p(\lambda)$. If the root of $p(\lambda)$ are distinct then

$$N_K(T) = \sum_{i=1}^{m} \alpha_i \left[\phi_i\right]^T = \sum_{i=1}^{m} |\alpha_i| \, |\phi_i|^T \cos(\theta_i(T))$$

where $\alpha_i = \dfrac{\phi_i^{m-1}}{p'(\phi_i)}$ and $\theta_i(t) = \arg(\alpha_i) + t \arg(\phi_i)$

Proof: Let M be the companion matrix. It is easy to show that

$$M = \begin{bmatrix} K_1 & K_2 & . & K_{m-1} & K_m \\ 1 & 0 & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & 0 & 0 \\ 0 & 0 & . & 1 & 0 \end{bmatrix}$$

For $i \in \{1, 2, ..., m-1, m\}$, we define C_i and R_i as follows,

$$C_i = \begin{bmatrix} \phi_i^{m-1} & \phi_i^{m-2} & . & \phi_i^1 & 1 \end{bmatrix}^T \text{ and}$$

$$R_i = \begin{bmatrix} 1 & \phi_i^1 - \sum_{j=1}^{1} K_j \frac{\phi_i^1}{\phi_i^j} & . & . & \phi_i^{m-1} - \sum_{j=1}^{m-1} K_j \frac{\phi_i^{m-1}}{\phi_i^j} \end{bmatrix}^T$$

Since $p(\phi_i) = 0$, $\phi_i^m - \sum_{i=1}^{m-1} K_i \phi_i^{m-i} = K_m$ and $\phi_i^m = \sum_{i=1}^{m} K_i \phi_i^{m-i}$. Using these two identities, it is easy to show that $MC_i = \phi_i C_i$ and $M^T R_i = \phi_i R_i$. Thus C_i and R_i are the column and row eigenvectors of M.

Next we will establish that if $i \neq j$, then $R_i^T C_j = 0$. $(R_i^T M) C_j = R_i^T (M C_j)$. But $(R_i^T M) C_j = \phi_i R_i^T C_j$ and $R_i^T (M C_j) = \phi_j R_i^T C_j$. Hence, $\phi_i R_i^T C_j = \phi_j R_i^T C_j$. Hence, $\phi_i = \phi_j$. This is a contradiction, since the roots of the characteristic polynomial are distinct.

Since, the roots of the characteristic polynomial are distinct and since $R_i C_j = 0$ if $i \neq j$, using an argument similar to the one by Capocelli and Cull [9], it is possible to show that

$$N_K(T) = \sum_{i=1}^{m} \alpha_i [\phi_i]^T$$

where $\alpha_i = \dfrac{R_i^T I}{\phi_i R_i^T C_i}$ and

$$I = \begin{bmatrix} N_K(m) & N_K(m-1) & . & N_K(2) & N_K(1) \end{bmatrix}^T$$

To complete the derivation, we need to establish the exact value of $R_i^T I$ and $R_i^T C_i$. First let us derive an expression for $R_i^T I$.

$$R_i^T I = N_K(m) + \sum_{j=1}^{m-1} [\phi_i^j - \sum_{l=1}^{j} K_l \phi_i^{j-l}] N_K(m-j)$$

$$= N_K(m) + \sum_{j=1}^{m-1} \phi_i^j N_K(m-j) - \sum_{j=1}^{m-1} \sum_{l=1}^{j} K_l \phi_i^{j-l} N_K(m-j)$$

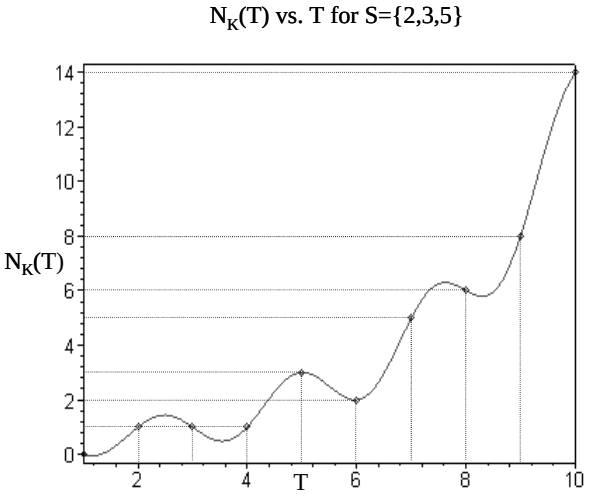


Fig. 8.1. $N_K(T)$ vs. T for $S = \{2, 3, 5\}$. $N_K(T)$ oscillates and increases exponentially.

Collecting like powers of ϕ_i

$$= N_K(m) + \sum_{p=1}^{m-1} \phi_i^p N_K(m-p) - \sum_{p=0}^{m-2} \phi_i^p \sum_{j=1}^{m-p-1} K_j N_K(m-p-j)$$

$$= [N_K(m) - \sum_{j=1}^{m-1} K_j N_K(m-j)] + [\phi_i^{m-1} N_K(1)] + \sum_{p=1}^{m-2} \phi_i^p [N_K(m-p) - \sum_{j=1}^{m-p-1} K_j N_K(m-p-j)]$$

Now note that, $N_K(m) = K_m + \sum_{j=1}^{m-1} K_j N_K(m-j)$

Hence, $N_K(m) - \sum_{j=1}^{m-1} K_j N_K(m-j) = K_m$

Similarly, $N_K(m-p) - \sum_{j=1}^{m-p-1} K_j N_K(m-p-j) = K_{m-p}$

Also $N_K(1) = K_1$. Substituting these values into the formula for $R_i^T I$, we get

$$R_i^T I = K_m + \phi_i^{m-1} K_1 + \sum_{p=1}^{m-2} \phi_i^p K_{m-p} = \sum_{j=1}^{m} K_j \phi_i^{m-j}$$

But since $p(\phi_i) = 0$, $\phi_i^m = \sum_{j=1}^{m} K_j \phi_i^{m-j}$

Hence, $R_i^T I = \phi_i^m$

Now we will derive an expression for $R_i^T C_i$.

$$R_i^T C_i = \phi_i^{m-1} + \sum_{n=1}^{m-1} [\phi_i^n - \sum_{j=1}^{n} K_j \phi_i^{n-j}] \phi_i^{m-1-n}$$

$$= \phi_i^{m-1} + \sum_{n=1}^{m-1} \phi_i^{m-1} - \sum_{n=1}^{m-1} \sum_{j=1}^{n} K_j \phi_i^{m-1-j}$$

$$= m\phi_i^{m-1} - \sum_{j=1}^{m-1} K_j (m-j) \phi_i^{m-j-1}$$

$$= m\phi_i^{m-1} - \sum_{j=1}^{m} K_j (m-j) \phi_i^{m-j-1} = p'(\phi_i)$$

Now we are ready to derive the value of the multiplicative constant. It follows by substituting the value of $R_i^T I$ and $R_i^T C_i$ in the formula for α_i.

$$\alpha_i = \frac{R_i^T I}{\phi_i R_i^T C_i} = \frac{\phi_i^{m-1}}{p'(\phi_i)}$$

The characteristic polynomial can have complex roots. Thus we will derive the expression for $N_K(T)$ in polar form. First let us define

$$\theta_i(t) = \arg(\alpha_i) + t \arg(\phi_i)$$

$$N_K(T) = \sum_{i=1}^{m} |\alpha_i| \, |\phi_i|^T \left[\cos(\theta_i(T)) + \sqrt{-1} \sin(\theta_i(T))\right]$$

Note that the complex roots occur in conjugate pairs. It is easy to show that if ϕ_i and ϕ_j are conjugates, so are α_i and α_j. Hence, $\theta_i(T) = -\theta_j(T)$. Thus the complex sinusoid components of the i and j term in the summation will cancel. Thus,

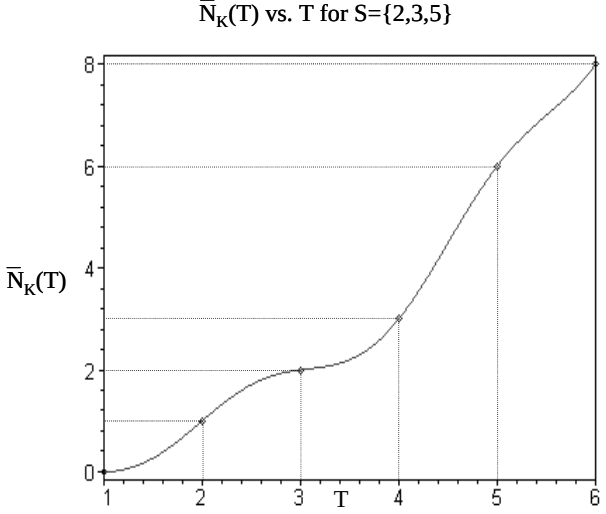


Fig. 8.2. $\bar{N}_K(T)$ vs. T for $S = \{2, 3, 5\}$. $\bar{N}_K(T)$ oscillates and increases exponentially.

$$N_K(T) = \sum_{i=1}^{m} |\alpha_i| \, |\phi_i|^T \cos(\theta_i(T)) \qquad \blacksquare$$

We define the sum of the Generalized Fibonacci Numbers to be $\bar{N}_K(T) = \sum_{t=1}^{T} N_K(t)$. Before we derive a closed form expression for the sum of the Generalized Fibonacci Numbers, we will prove a lemma. This lemma is similar to a lemma proven by Mukhtar and Bruck [37].

Lemma 8.1: $\sum_{i=1}^{m} \frac{\alpha_i \phi_i}{\phi_i - 1} = \frac{\sum_{i=1}^{m} K_i}{[\sum_{i=1}^{m} K_i] - 1}$

Proof: First let us define a new function G as follows

$$G_K(n) = \begin{cases} \sum_{i=1}^{m} K_i G_K(n-i) & n > m \\ \sum_{i=n}^{m} K_i + \sum_{i=1}^{n-1} K_i G_K(n-i) & 2 \le n \le m \\ \sum_{i=1}^{m} K_i & n = 1 \end{cases}$$

Next we will prove

$G_K(n) = \sum_{i=1}^{m} \gamma_i \phi_i^n$

where $\gamma_i = ([\sum_{j=1}^{m} K_j] - 1)\alpha_i \, (\phi_i - 1)^{-1}$

Since, G is a generalized Fibonacci type recurrence having the same characteristic polynomial, it can be written as,

$G_K(n) = \sum_{i=1}^{m} \gamma_i \phi_i^n$.

The only issue that needs to be addressed is the exact value of γ_i. Using an argument similar to that in Capocelli and Cull [9], it is easy to show that,

$\gamma_i = \frac{R_i^T I}{\phi_i R_i^T C_i} = \frac{R_i^T I}{\phi_i p'(\phi_i)}$ where

$$I = \begin{bmatrix} G_K(m) & G_K(m-1) & . & G_K(2) & G_K(1) \end{bmatrix}^T$$

$(\phi_i - 1) \, R_i^T I$

$= (\phi_i - 1) \left[G_K(m) + \sum_{n=1}^{m-1} (\phi_i^n - \sum_{j=1}^{n} K_j \phi_i^{n-j}) G_K(m-n) \right]$

By collecting like powers of ϕ_i we get,

$= (\phi_i - 1) \sum_{p=0}^{m-1} \phi_i^p \left[G_K(m-p) - \sum_{j=1}^{m-p-1} K_j G_K(m-p-j) \right]$

But we know that,

$G_K(m-p) - \sum_{j=1}^{m-p-1} K_j G_K(m-p-j) = \sum_{j=m-p}^{m} K_j$

Substituting,

$= (\phi_i - 1) \sum_{p=0}^{m-1} (\phi_i^p \sum_{j=m-p}^{m} K_j)$

$= \sum_{p=0}^{m-1} (\phi_i^{p+1} \sum_{j=m-p}^{m} K_j) - \sum_{p=0}^{m-1} (\phi_i^p \sum_{j=m-p}^{m} K_j)$

$= \sum_{p=1}^{m} (\phi_i^p \sum_{j=m-p+1}^{m} K_j) - \sum_{p=0}^{m-1} (\phi_i^p \sum_{j=m-p}^{m} K_j)$

$= \phi_i^m \sum_{j=1}^{m} K_j + \sum_{p=1}^{m-1} \phi_i^p (\sum_{j=m-p+1}^{m} K_j - \sum_{j=m-p}^{m} K_j) - K_m$

$= \phi_i^m \sum_{j=1}^{m} K_j - \sum_{p=1}^{m-1} \phi_i^p K_{m-p} - K_m$

$= \phi_i^m \sum_{j=1}^{m} K_j - \sum_{j=1}^{m} K_j \phi_i^{m-j} = \phi_i^m ([\sum_{j=1}^{m} K_j] - 1)$

We have shown that,

$(\phi_i - 1) \, R_i^T I = \phi_i^m ([\sum_{j=1}^{m} K_j] - 1)$

Hence,

$R_i^T I = \frac{1}{(\phi_i - 1)} \phi_i^m ([\sum_{j=1}^{m} K_j] - 1)$

Substituting we get,

$\gamma_i = \frac{R_i^T I}{\phi_i p'(\phi_i)} = ([\sum_{j=1}^{m} K_j] - 1) \frac{\phi_i^{m-1}}{(\phi_i - 1) \, p'(\phi_i)}$

Thus we have shown,

$G_K(n) = \sum_{i=1}^{m} \gamma_i \phi_i^n$

where $\gamma_i = ([\sum_{j=1}^{m} K_j] - 1)\alpha_i \, (\phi_i - 1)^{-1}$

Now note that,

$G_K(1) = \sum_{i=1}^{m} K_i$

Also,

$G_K(1) = \sum_{i=1}^{m} \gamma_i \phi_i = ([\sum_{i=1}^{m} K_i] - 1)(\sum_{i=1}^{m} \frac{\alpha_i \phi_i}{\phi_i - 1})$

Hence,

$\sum_{i=1}^{m} \frac{\alpha_i \phi_i}{\phi_i - 1} = \frac{\sum_{i=1}^{m} K_i}{[\sum_{i=1}^{m} K_i] - 1}$ $\blacksquare$

Next we will derive a closed form expression for the sum of the Generalized Fibonacci Numbers.

Theorem 8.2: Let $p(\lambda) = \lambda^m - \sum_{i=1}^{m} K_i \lambda^{(m-i)}$ and $p'(\lambda) = m\lambda^{m-1} - \sum_{i=1}^{m} K_i(m-i)\lambda^{(m-i-1)}$. For $i \in \{1, 2, ..., m-1, m\}$ let ϕ_i be the roots of $p(\lambda)$. If the roots of $p(\lambda)$ are distinct then

$$\bar{N}_K(T) = \kappa + \sum_{i=1}^{m} \beta_i [\phi_i]^{T+1}$$
$$= \kappa + \sum_{i=1}^{m} |\beta_i| |\phi_i|^{T+1} \cos(\theta_i(T+1))$$

where $\beta_i = \dfrac{\phi_i^{m-1}}{(\phi_i - 1)p'(\phi_i)}$,
$\theta_i(t) = \arg(\beta_i) + t \arg(\phi_i)$ and
$\kappa = -\dfrac{\sum_{i=1}^{m} K_i}{[\sum_{i=1}^{m} K_i] - 1}$

Proof: $\bar{N}_K(T) = \sum_{t=1}^{T} N_K(T) = \sum_{t=1}^{T} \sum_{i=1}^{m} \alpha_i [\phi_i]^t$
$= \sum_{i=1}^{m} \alpha_i \sum_{t=1}^{T} [\phi_i]^t = \sum_{i=1}^{m} \alpha_i \dfrac{\phi_i^{T+1} - \phi_i}{\phi_i - 1}$
$= -\sum_{i=1}^{m} \alpha_i \dfrac{\phi_i}{\phi_i - 1} + \sum_{i=1}^{m} \alpha_i \dfrac{\phi_i^{T+1}}{\phi_i - 1}$

But note that by Lemma 8.1,

$$\sum_{i=1}^{m} \alpha_i \frac{\phi_i}{\phi_i - 1} = \frac{\sum_{i=1}^{m} K_i}{[\sum_{i=1}^{m} K_i] - 1} = -\kappa$$

Hence, $\bar{N}_K(T) = \kappa + \sum_{i=1}^{m} \alpha_i \dfrac{\phi_i^{T+1}}{\phi_i - 1} = \kappa + \sum_{i=1}^{m} \beta_i \phi_i^{T+1}$

Now we convert this formula to polar form. First we define,

$$\theta_i(t) = \arg(\beta_i) + t \arg(\phi_i)$$
$$\bar{N}_K(T) = \kappa + \sum_{i=1}^{m} |\beta_i| |\phi_i|^{T+1} [\cos(\theta_i(T+1)) + \sqrt{-1} \sin(\theta_i(T+1))]$$

Again note that the complex roots occur in conjugate pairs. It is easy to show that if ϕ_i and ϕ_j are conjugates, so are β_i and β_j. Hence, $\theta_i(T) = -\theta_j(T)$. Thus the complex sinusoid components of the i and j term in the summation will cancel. Thus,

$$\bar{N}_K(T) = \kappa + \sum_{i=1}^{m} |\beta_i| |\phi_i|^{T+1} \cos(\theta_i(T+1)) \quad \blacksquare$$

Definition 8.1: We will formally define the mth order Fibonacci Numbers F and their sum $\bar{F}$ as

$$F_m(n) = \begin{cases} \sum_{i=1}^{m} F(n-i) & n \geq m \\ 2^{n-2} & 2 \leq n \leq m-1 \\ 1 & n = 1 \\ 0 & n = 0 \end{cases}$$

and $\bar{F}(n) = \sum_{i=1}^{n} F(i)$

We will leave as an exercise for the reader to verify that when $S = \{1, 2, ..., m-1, m\}$ or equivalently $K_i = 1$ for all $i \in \{1, 2, ..., m-1, m\}$ then $F_m(T) = N_K(T-1)$ and $\bar{F}_m(T) = \bar{N}_K(T-1) + 1$. We let $p(\lambda) = \lambda^m - \sum_{i=1}^{m} \lambda^{(m-i)}$. Also let ϕ_i be the roots of $p(\lambda) = 0$ (it is easy to verify that the roots are distinct) and let ϕ_1 be the real positive root (it is easy to verify that the real positive root is unique). From Theorem 8.1 it follows that

$$F_m(T) = \sum_{i=1}^{m} \alpha_i [\phi_i]^{T-1} \text{ where } \alpha_i = \frac{\phi_i^{m-1}}{p'(\phi_i)}$$

Capocelli and Cull [9] further show that the mth order Fibonacci Numbers are rounded powers and can be computed from only the real positive root using the formula below.

$$F_m(T) = \left\langle \alpha_1 \phi_1^{T-1} \right\rangle$$

From Theorem 8.2 it follows that

$$\bar{F}_m(T) = -\frac{1}{m-1} + \sum_{i=1}^{m} \beta_i [\phi_i]^T \text{ where } \beta_i = \frac{\phi_i^{m-1}}{(\phi_i - 1)p'(\phi_i)}$$

In [37] we establish that the sum of the Fibonacci Numbers, like the Fibonacci Numbers are rounded powers and can be computed using the formula below.

$$\bar{F}_m(T) = \left\langle -\frac{1}{m-1} + \beta_1 \phi_1^T \right\rangle$$

IX. Conclusions and Future Work

In this paper we have introduced a new paradigm for communication and storage which can potentially provide substantial improvements in terms of rate and storage density. We call this paradigm Interval Modulation Coding. Both in the context of communication and storage, one needs to measure elapsed time between voltage transitions or voltage pulses and conventionally this measurement is made by a clock based circuit, by counting clock pulses. The key idea is that one can use analog circuits (or clocks of higher frequency) to measure elapsed time and the set of permissible time intervals must be chosen in accordance with a probabilistic model of measurement error.

Given a set of permissible time intervals, we have provided a mechanism for encoding and decoding binary data based on variable length to variable length prefix-free codes. We show that such codes can be constructed using integer linear programming. From a theoretical standpoint, we study the linear programming relaxation of the integer linear program. We provide an efficient algorithm for determining if the linear programming relaxation of the integer program has a solution, and an efficient algorithm for determining the optimal solution if one exists. Furthermore, we have derived closed form expressions for the Generalized Fibonacci Numbers and their sums. This work is an extension of the work of Capocelli and Cull [9].

Our current work focuses on developing an efficient procedure for solving the integer linear program (with the integrality constraint) associated with code construction. Also we are developing integrated transceivers for Fiber Optic Communication based on the principles of Interval Modulation Coding outlined in this paper. This work is being done collaboratively with Behnam Analui and Ali Hajimiri at Caltech. Using these techniques we hope to surpass the data rates achieved by Greshishchev et. al. [19].

Acknowledgments

We would like to thank Brian Marcus for providing many references on the topic of run length limited coding and Ken Zeger for reviewing early drafts of this paper.

References

[1] R. L. Adler, D. Coppersmith and M. Hassner, "Algorithms for Sliding Block Codes – an Application of Symbolic Dynamics to Information Theory", IEEE Transactions on Information Theory, vol. 29, pp. 5-22, 1983.

[2] D. Altenkamp and K. Mehlhorn, "Codes: Unequal Probabilities, Unequal Letter Costs", J. ACM, vol. 27, no. 3, pp. 412-427, July 1980.

[3] E. Arikan, "An Implementation of Elias Coding for Input-Restricted Channel", IEEE Transactions on Information Theory, vol. 36, pp. 162-165, 1990.

[4] M. S. Bazaraa, J J. Jarvis and H. D. Sherali, "Linear Programming and Network Flows", John Wiley & Sons, Inc., 1977, 1990.

[5] N. M. Blachman, "Minimum-Cost Encoding of Information", IRE Trans. Inform. Theory, vol. PGIT-3, pp. 139-149, 1954.

[6] M. Blaum and J. Bruck, "Coding for Skew Tolerant Parallel Asynchronous Communications", IEEE Transactions on Information Theory, vol. 39, no. 2, pp. 379-388, March 1993.

[7] M. Blaum, J. Bruck and L.H. Khachatrian, "Constructions of Skew-Tolerant and Skew-Detecting Codes", IEEE Transactions on Information Theory, vol. 39, no. 5, pp. 1752-1757, September 1993.

[8] M. Blaum and J. Bruck, "Coding for Delay-Insensitive Communication with Partial Synchronization", IEEE Transactions on Information Theory, vol. 40, no. 3, pp. 941-945, May 1994.

[9] R. M. Capocelli and P. Cull, "Generalized Fibonacci Numbers are Rounded Powers", Third International Conference on Fibonacci Numbers and Their Applications, Pisa, Italy, pp. 57-62, 1988.

[10] N. Cot, "Complexity of the variable length encoding problem", Proc. 6th Southeast Conference on Combinatorics, Graph Theory and Computing, Congressus Numerantium XIV, Utilitas Mathematica Publishing, Winnepeg, MB, Canada, pp. 211-244, 1975.

[11] N. Cot, "Characterization and Design of Optimal Prefix Codes", Ph.D. Dissertation, Stanford University, Stanford, CA., 1977.

[12] L. Farina and S. Rinaldi, "Positive Linear Systems – Theory and Applications", John Wiley & Sons, 2000.

[13] P. A. Franaszek, "Sequence-state Coding for Digital Transmission", Inform. Contr., vol. 1-J, pp. 155-164, 1969.

[14] P. Funk, "Run-length-limited Codes with Multiple Spacing", IEEE Transactions on Magnetics, vol. 18, pp. 772-775, 1982.

[15] F. R. Gantmacher, "Applications of the Theory of Matrices", Interscience Publishers, 1959.

[16] E. N. Gilbert, "Coding with Digits of Unequal Cost", IEEE Transactions on Information Theory, vol. 41, no. 2, pp. 596-600, 1995.

[17] M. J. Golin and N. Young, "Prefix Codes: Equiprobable Words, Unequal Letter Costs", SIAM J. Comput., vol. 25, no. 6, pp. 1281-1292, 1996.

[18] M. J. Golin and G. Rote, "A Dynamic Programming Algorithm for Constructing Optimal Prefix-Free Codes with Unequal Letter Costs", IEEE Transactions on Information Theory, vol. 44, no. 5, pp. 1770-1781, 1998.

[19] Y. M. Greshishchev, P. Schvan, J. L. Showell, M. Xu, J. J. Ojha and J. E. Rogers, "A Fully Integrated SiGe Receiver IC for 10-Gb/s Data Rate", IEEE Journal of Solid State Circuits, vol. 35, no. 12, pp. 1949-1957, Dec. 2000.

[20] C. D. Heegard, B. H. Marcus and P. H. Siegel, "Variable Length State Splitting with Applications to Average Runlength-constrained (ARC) Codes", IEEE Transactions on Information Theory, vol. 37, pp. 759-777, 1991.

[21] D. H. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the IRE, vol. 40, no. 9, pp. 1098-1101, 1952.

[22] K. A. S. Immink, "Codes for Mass Digital Storage", Shannon Foundation Publishers, 1999.

[23] S. Kapoor and E. M. Reingold, "Optimum Lopsided Binary Trees", J. ACM, vol. 36, pp. 573-590, 1989.

[24] R. M. Karp, "Minimum Redundancy Coding for the Discrete Noiseless Channel", IRE Trans. Inf. Theory", vol. 7, no. 1, pp. 27-38, 1961.

[25] W. H. Kautz, "Fibonacci Codes for Synchronization Control", IEEE Transactions on Information Theory, pp. 284-292, April 1965.

[26] K. J. Kerpez, "Runlength Codes from Source Codes", IEEE Transactions on Information Theory, vol. 37, pp. 682-687, 1991.

[27] D. E. Knuth, "Efficient Balanced Codes", IEEE Transactions on Information Theory, vol. 32, pp. 51-53, 1986.

[28] R. M. Krause, "Channels which transmit letters of unequal duration", Inf. Control, vol. 5, no.1, pp. 13-24, 1962.

[29] D. A. Lelewer and D. S. Hirschberg, "Data Compression", ACM Computing Surveys, vol. 19, no. 3, pp. 261-296, 1987.

[30] D. Lind and B. Marcus, "An Introduction to Symbolic Dynamics and Coding", Cambridge University Press, 1985.

[31] B. H. Marcus, P. H. Siegel and J. K. Wolf, "Finite-state Modulation Codes for Data Storage", IEEE J. Sel. Areas Comm., vol. 10, pp. 5-37, 1992.

[32] B. H. Marcus, P. H. Siegel and J. K. Wolf, "Codes with a Multiple Spectral Null at Zero Frequency", IEEE Transactions on Information Theory, vol. 35, pp. 463-472, 1989.

[33] B. H. Marcus, R. Roth and P. H. Siegel, "Handbook of Coding Theory", Elsevier Press, 1998.

[34] R. S. Marcus, "Discrete Noiseless Coding", M. S. Thesis, MIT, Electrical Engineering Dept., 1957.

[35] R. K. Martin, "Large Scale Linear and Integer Optimization", Kluwer Academic Publishers, 1999.

[36] K. Mehlhorn, "An Efficient Algorithm for Constructing Nearly Optimal Prefix Codes", IEEE Trans. Inf. Theory, vol. 26, no. 5, pp. 513-517, 1980.

[37] S. Mukhtar and J. Bruck, "Frequency Modulation for Asynchronous Data Transfer", Electronic Technology Report, ETR036, April, 2001.

[38] K. Murty, "Linear Programming", John Wiley & Sons, Inc., 1983.

[39] Y. Perl, M. R. Garey and S. Even, "Efficient Generation of Optimal Prefix Code: Equiprobable Words using Unequal Cost Letters", J. ACM, vol. 22, no. 2, pp. 202-214, 1975.

[40] A. Schrijver, "Theory of Linear and Integer Programming", John Wiley & Sons, Inc., 1986.

[41] E. Sperner, "Ein Satz uber Untermengen einer endlichen Menge", Math. Z. vol. 27, pp. 544-548, 1928.

[42] W. R. Spickerman, "Binet's Formula for the Tribonacci Sequence", The Fibonacci Quarterly, no. 2, pp. 118-120, May 1982.

[43] W. R. Spickerman and R. N. Joyner, "Binet's Formula for the Recursive Sequence of Order K", The Fibonacci Quarterly, no. 4, pp. 327-331, 1984.

[44] G. Strang, "Introduction to Linear Algebra", Wellesley-Cambridge Press, 1993.

[45] B. P. Tunstall, "Synthesis of Noiseless Compression Codes", Thesis Georgia Institute of Technology, 1967.

[46] B. Varn, "Optimal Variable Length Codes (Arbitrary Symbol Cost and Equal Codeword Probability)", Inf. Control. vol. 19, no. 4, pp. 289-301, 1971.

[47] T. Verhoeff, "Delay-insensitive codes – an overview", Distributed Computing, pp. 3:1-8, 1988.

Appendix

Notation 10.1: S is the set of permissible time intervals and we assume it consists of positive integers and has at least two elements

Σ_S^+ is the set of non-null strings over S
For all $s \in \Sigma_S^+$, $\|s\|$ is the sum of the elements in s
$\Pi_S^l = \{s | s \in \Sigma_S^+ \text{ and } \|s\| = l\}$
Assume $p \in \Sigma_S^+$ and $l \geq \|p\|$. We define $P_A(p, l) = \{s | s \in \Sigma_S^+ \text{ and } \|s\| = l \text{ and } p \text{ is a prefix of } s\}$

Theorem 10.1: We are given $M_S \subseteq \Sigma_S^+$ such that M_S is a prefix-free set. For all i let $y_i = |\{s | s \in M_S \text{ and } \|s\| = i\}|$.
For all n, $y_n \leq N_K(n) - \sum_{l=1}^{n-1} N_K(n-l) y_l$
Furthermore, if we are given y_i such that they satisfy the constraints above, we can find $M_S \subseteq \Sigma_S^+$ such that M_S is a prefix-free set and $|\{s | s \in M_S \text{ and } \|s\| = i\}| = y_i$.

Proof: We are given n. Define $\bar{M}_S(n) = \{s | s \in M_S \text{ and } \|s\| < n\}$ and $\bar{R}_S(n) = \{s | s \in M_S \text{ and } \|s\| = n\}$.

Note that, $\bar{R}_S(n) \cup \bigcup_{s \in \bar{M}_S(n)} P_S(s, n) \subseteq \Pi_S^n$

Hence, $\left| \bar{R}_S(n) \cup \bigcup_{s \in \bar{M}_S(n)} P_S(s, n) \right| \leq |\Pi_S^n|$

It is easy to show that $\bar{R}_S(n) \cap \bigcup_{s \in \bar{M}_S(n)} P_S(s, n) = \phi$ and since $\bar{M}_S(n)$ is a prefix-free set for all s_1, $s_2 \in \bar{M}_S(n)$, $P_S(s_1, n) \cap P_S(s_2, n) = \phi$.

Hence, $\left| \bar{R}_S(n) \right| + \sum_{s \in \bar{M}_S(n)} |P_S(s, n)| \leq |\Pi_S^n|$

Hence, $y_n \leq N_K(n) - \sum_{l=1}^{n-1} N_K(n-l) y_l$

The proof of the converse is constructive. Note that $y_1 \leq N_K(1) = K_1$. If $K_1 = 0$, let $M_S = \{\}$ and if $K_1 = 1$, let $M_S = \{1\}$. Now we will assume that we have constructed a prefix-free set such that for $i \in \{1, ..., n-1\}$, $|\{s | s \in M_S \text{ and } \|s\| = i\}| = y_i$.

Note that, $y_n \leq N_K(n) - \sum_{l=1}^{n-1} N_K(n-l) y_l$

Now let $U = \Pi_S^n - \bigcup_{s \in M_S} P_S(s, n)$

Hence, $|U| = \left| \Pi_S^n - \bigcup_{s \in M_S} P_S(s, n) \right|$

First note that $\bigcup_{s \in M_S} P_S(s, n) \subseteq \Pi_S^n$. Also note that since $M_S(n)$ is a prefix-free set for all s_1, $s_2 \in M_S(n)$, $P_S(s_1, n) \cap P_S(s_2, n) = \phi$.

Hence, $|U| = N_K(n) - \sum_{l=1}^{n-1} N_K(n-l) y_l$

Hence, $y_n \leq |U|$. It suffices to pick y_n elements from U and add them to M_S. ■